

1.50
1.95
Road
MA 01901

MICRO

THE 6502/6809 JOURNAL

TM P.70: SPEECH
P.73: AIN
RUPAIR
P.83: WWS BOARD

P.35 BUL BOARD



Commodore Feature

ATARI Graphics

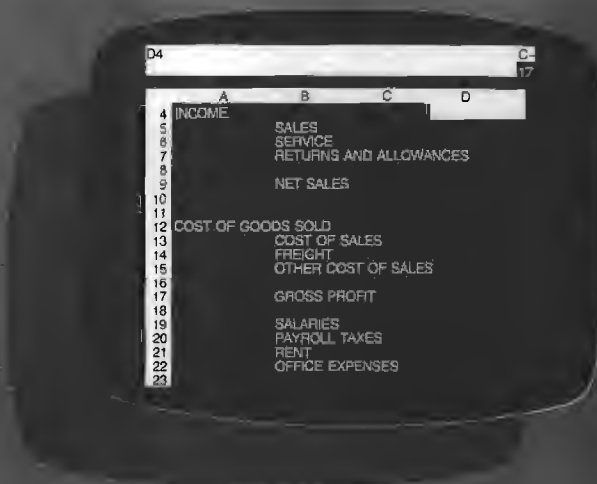
APPLESOFT GOTO/GOSUB Checker

68000 Logic Instructions



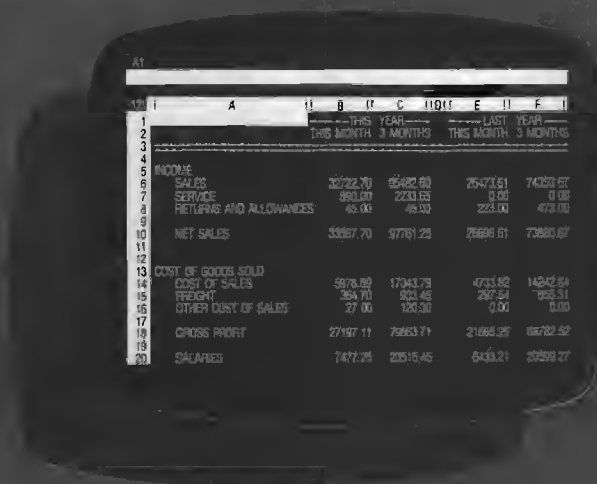
MAGICALC™

The Visual Difference



VisiCalc™

	A	B	C	D
4	INCOME			
5		SALES		
6		SERVICE		
7		RETURNS AND ALLOWANCES		
8				
9		NET SALES		
10				
11				
12		COST OF GOODS SOLD		
13		COST OF SALES		
14		FREIGHT		
15		OTHER COST OF SALES		
16				
17		GROSS PROFIT		
18				
19		SALARIES		
20		PAYROLL TAXES		
21		RENT		
22		OFFICE EXPENSES		
23				



MAGICALC

	A	B	C	D	E	F
1		THIS YEAR		LAST YEAR		
2		THIS MONTH	3 MONTHS	THIS MONTH	3 MONTHS	
3						
4						
5		INCOME				
6		SALES	32722.70	15482.80	25471.51	74253.67
7		SERVICE	880.00	223.55	0.00	0.00
8		RETURNS AND ALLOWANCES	45.00	45.00	223.00	473.00
9						
10		NET SALES	33657.70	15660.35	25668.51	73880.67
11						
12						
13		COST OF GOODS SOLD				
14		COST OF SALES	5976.50	17943.75	4733.82	14242.84
15		FREIGHT	354.70	821.45	267.54	555.31
16		OTHER COST OF SALES	27.00	120.00	0.00	0.00
17						
18		GROSS PROFIT	27197.11	7963.71	21066.95	60722.82
19						
20		SALARIES	7477.25	23515.45	5433.21	20299.27

You Won't Need VisiCalc™ To See Your Savings

MAGICALC is the second generation of spreadsheet programs. A state-of-the-art spreadsheet program, it provides you with all the power you need by fully utilizing the most recent software and hardware breakthroughs—without your having to break the bank to obtain them.

MAGICALC includes:

- 70-column upper- and lower-case video display
- Full 80-column board display
- 40-column standard display
- Individual column widths
- Invisible columns for bar graphs and charts
- Full compatibility with VisiCalc
- Hard-disk compatibility
- Special offer for VisiCalc owners
- Magic editing key

MAGICALC™—An outstanding product at a surprisingly low price

\$149⁹⁵

Breakthrough to today's technology by contacting:

5547 satsuma avenue • north hollywood, california 91601 • 213/985-2922

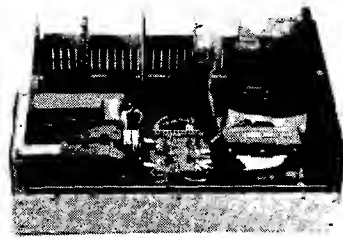




FLEX - OS-9 LEVEL ONE - UNIFLEX - OS-9 LEVEL TWO

ONLY GIMIX Systems can be configured to run any of these.

GIMIX systems utilize the most powerful 6809 operating systems: FLEX, UniFLEX, OS-9 LEVEL ONE and TWO -- the systems the PROs use. This means a wide selection of software to choose from as well the ability to develop sophisticated, multi-user/multi-tasking programs on your GIMIX System.



The GIMIX CLASSY CHASSIS™ consists of a heavy-weight aluminum mainframe cabinet which provides more than ample protection for the electronics and 1 or 2 optional 5 1/4" drives.

Backpanel connectors can be added for convenient connection of terminals, printers, drives and other peripherals.

A 3 position locking keyswitch enables users to disable the front panel reset button to prevent accidental or unauthorized tampering with the system.

The GIMIX system mother board provides fifteen 50 pin slots and eight 30 pin I/O slots -- the most room for expansion of any SS50 system available. The on board baud rate generator features 11 standard baud rates, 75 to 38.4K, for maximum versatility and compatibility with other systems. Extended address decoding allows the I/O block to be addressed anywhere in the 1 megabyte address space. All components feature Gold plated connectors for a lifetime of solid connections. All boards are fully buffered for maximum system expansion.

Each GIMIX Mainframe System is equipped with an industrial quality power supply featuring a **ferro-resonant constant voltage transformer** to insure against problems caused by adverse power input conditions such as A.C. line voltage fluctuations etc. The supply provides 8 volts at 30 amps and plus or minus 16 volts at 5 amps, more than enough capacity to power a fully loaded system and two internal drives.

The 2MHz GIMIX 6809 PLUS CPU board includes a time of day clock with battery back-up and 6840 programmable timer to provide the programmer with convenient, accurate time reference. Later addition of 9511 or 9512 arithmetic processors is provided for on the board. The unique GIMIX design enables software selection of either OS-9 or FLEX, both included in many complete GIMIX systems.

GIMIX STATIC RAM boards require no complicated refresh timing cycles or clocks for data retention. GIMIX memory boards are guaranteed for 2 MHz operation with no wait state or clock stretching required.

Our low power NMOS RAM requires less than 3/4 amp at 8V for a fully populated 64K board. For critical situations, our non-volatile 64K byte CMOS static RAM boards with built in battery back-up retain data even with system power removed. A fully charged battery will power this board for a minimum of 21 days. A write protect switch permits CMOS boards to be used for PROM/ROM emulation and software debugging.

The GIMIX DMA controller leaves the processor free to perform other tasks during disk transfers - an important feature for multi-user/multi-tasking systems where processor time allocation is critical. The DMA board will accommodate up to 4 drives 5 1/4" or 8" in any combination running single or double density single or double headed. Programmed I/O Disk Controllers are also available.

GIMIX systems are designed with ultimate **RELIABILITY** in mind. You can choose from the below featured systems or select from our wide variety of components to build a custom package to suit your needs.

GIMIX 2MHz 6809 System including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 56KB STATIC RAM, 2 SERIAL PORTS W/CABLES, GMXBUG MONITOR, FLEX, and OS-9 LEVEL 1 **\$3248.49**

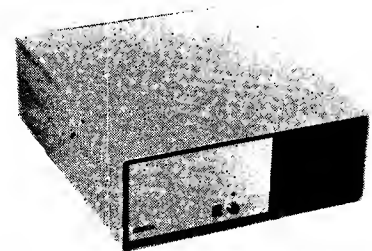
FOR TWO 5 1/4" 40 TRACK DSDD DRIVES ADD **\$ 900.00**

GIMIX 128KB WINCHESTER SYSTEM including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 128KB STATIC RAM, 4 SERIAL PORTS W/CABLES, 5 1/4" 80 TRACK DSDD FLOPPY DISK DRIVE, 19MB 5 1/4" WINCHESTER HARD DISK, OS9 LEVEL 2, EDITOR AND ASSEMBLER **\$8998.09**

50HZ Versions Available, 8" Drives Available — Contact GIMIX for Prices and Information.

The Sun Never Sets On A GIMIX!

GIMIX users are found on every continent, including Antarctica. A representative group of GIMIX users includes: **Government Research and Scientific Organizations** in Australia, Canada, U.K. and in the U.S.; NASA, Oak Ridge, White Plains, Fermilab, Argonne, Scripps, Sloan Kettering, Los Alamos National Labs, AURA. **Universities:** Carleton, Waterloo, Royal Military College, in Canada; Trier in Germany; and in the U.S.; Stanford, SUNY, Harvard, UCSD, Mississippi, Georgia Tech. **Industrial users** in Hong Kong, Malaysia, South Africa, Germany, Sweden, and in the U.S.; GTE, Becton Dickinson, American Hoechst, Monsanto, Allied, Honeywell, Perkin Elmer, Johnson Controls, Associated Press, Aydin, Newkirk Electric, Revere Sugar, HI-G/AMS Controls, Chevron. **Computer mainframe and peripheral manufacturers,** IBM, OKI, Computer Peripherals Inc., Qume, Floating Point Systems. **Software houses;** Microware, T.S.C., Lucidata, Norpak, Talbot, Stylo Systems, AAA, HHH, Frank Hogg Labs, Epstein Associates, Softwest, Dynasoft, Research Resources U.K., Microworks, Meta Lab, Computerized Business Systems.



GIMIX Inc. reserves the right to change pricing and product specifications at any time without further notice.

GIMIX® and GHOST® are registered trademarks of GIMIX Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants Inc.
OS-9 is a trademark of Microware Inc.

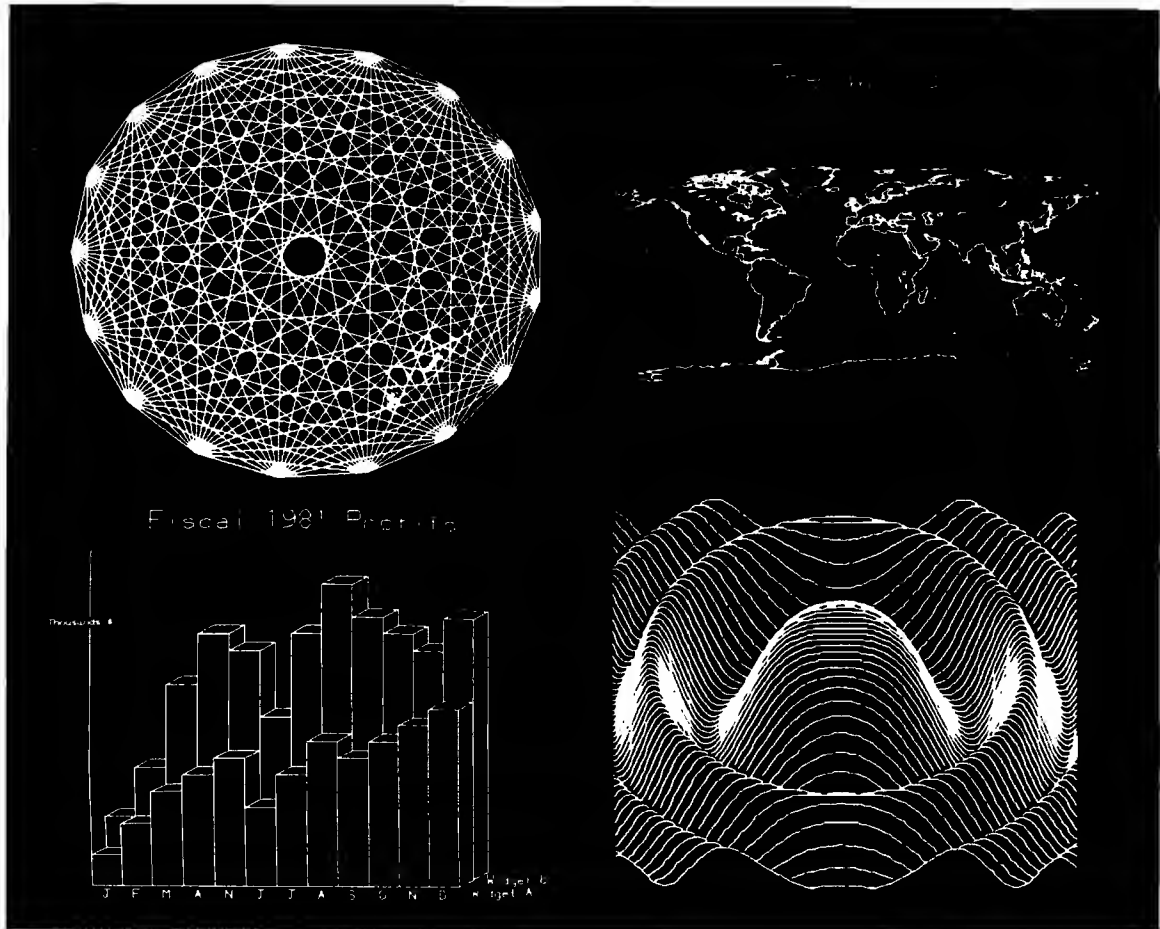
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

GIMIX Inc.

*The Company that delivers
Quality Electronic products since 1975.
© 1982 GIMIX Inc.*

ANNOUNCING **ElectroScreen™** the Superior Alternative to the Traditional Alphanumeric Terminals

only
\$595



The ElectroScreen™ Intelligent Graphics Board Features:

Graphics

- 512 x 480 resolution bit-mapped display
- Interleaved memory access — fast, snow-free updates

Intelligence

- 6809 on-board mpu
- 6K on-board firmware
- STD syntax high level graphics command set
- Removes host graphics software burden
- Flexible text and graphics integration
- Multiple character sizes
- User programs can be run on-board

Terminal

- Terminal emulation on power-up
- 83 characters by 48 lines display
- Easy switching among user-defined character sets
- Fast hardware scrolling

Additional Features

- SS-50C and SS-64 compatible board
- Board communicates with host through parallel latches
- Composite and TTL level video output
- 8 channel 8 bit A/D converter
- Board occupies 4 address bytes

See your dealer today!

The ElectroScreen manual is available for \$10, credited toward purchase of the board.

The ElectroScreen has a 90 day warranty from purchase date.

Dealers, please contact us for our special introductory package.



Privac Inc. (703) 671-3900
3711 S. George Mason Dr., Falls Church, Va. 22041

December Highlights

Commodore Machines Featured

This month we cover the full range of Commodore's machines: the PET, VIC, SuperPET, and the exciting new Commodore 64. Each machine has its own distinct features, but also shares characteristics with the other Commodore family members. CBM users will want to read all the Commodore related articles in this issue.

The second part of the University of Rochester's series (p. 59) discusses the use of an inexpensive device, the analog transducer, which can be applied to many problems outside the college teaching laboratory. The analog transducer makes it possible for your digital computer to deal with quantities measured on a continuous scale — light, voltages, densities.

Contributing Editor Jim Strasma starts on a six-part series (p. 37) that will help you write better program packages. In particular, it will cover CBM's powerful, yet poorly understood, relative record system. The first part, however, deals with designing a modular program package, setting things up, and passing parameters. Jim uses portions of the public domain program "Bennett's Mail List 4040" to illustrate his points.

We also offer a number of utilities for Commodore machines. Hans Hoogstraat's "BASIC Squeeze for PET" (p. 42) is a cassette buffer-sized program that can be saved with a fully expanded and commented BASIC program. When the program is run, it makes a call to the squeeze routine, which compresses the program to take less space and run faster. Troup and Strasma's "SOUP" (p. 52) is a compare program for machine-language routines saved on disk. Thomas Henry's "BASIC Line Delete for PET and VIC" (p. 47) adds the capability of deleting more than one BASIC program line at a time.

In our "Short Subjects" section (p. 97) we have two items of interest to users of Commodore machines. Terry Peterson explains the ASCII character set on the SuperPET and reveals some hidden features. "VIC Jitter Fixer," by Contributing Editor Dave Malmberg, can be added to your paddle, joystick, and light-pen programs to give you more reliable readings from these devices.

Finally, we feature the new Commodore 64 computer in both "PET Vet" and on our data sheet. Loren Wright's column (p. 54) reviews the graphic capabilities of this exciting new computer, and the data sheep (p. 109) provides a memory map, interfacing information, and lists of graphics and sound registers.

Expand Your Computer's Capabilities with New Hardware

The BSR X-10 allows you to control remotely a wide variety of electrical devices in your home. There are two versions available; one sends its signals using power lines as antennas, and another uses ultrasonic signals. Each light or appliance is connected to its own receiver module. John Krout's "Home Control Interface for C1P" (p. 77) shows how to add ultrasonic circuitry to your computer at a cost much less than the BSR ultrasonic option. David Hayes's "Atari Meets the BSR X-10" (p. 82) shows how to convert the unit for control from Atari's controller ports.

If you've ever looked at a 6502 programming manual, you might have noticed all the unused op codes. Now you can use those codes to execute your own machine-language routines. Curt Nelson and his associates ("Utilizing 6502's Undefined Operations," p. 93) present a circuit that causes the 6502 to execute your code, instead of crashing, when it encounters an unused op code.

In "Programmable Character Generator for OSI" Colin Macauley demonstrates how to define your own characters (p. 88). OSI readers should turn to our OSI book announcement on page 25.

Joe Hootman's in-depth coverage of the 68000's instruction set continues (p. 85) with a discussion of the logic instructions. As usual, convenient reference tables are included.

Apple and Atari

Paul Swanson concludes his three-part series on Atari's character graphics (p. 22) with a demonstration of patching into Atari's vertical blank interrupt routine. His "From Here to Atari" column (p. 32) covers a variety of topics, including Atari's new software acquisition centers and some technical tidbits.

Peter Meyer presents an "Applesoft GOTO/GOSUB Checking Routine" (p. 26) that displays all incorrect GOTO and GOSUB references. "ILISZT for Integer BASIC," by Leonard Anderson, is a follow up to a similar program he presented for Applesoft (p. 13). It produces an attractive, formatted listing of your Integer BASIC program, complete with indentation, paging, and other fancy features. Tim Osborn's "Apple Slices" (p. 65) presents a general-purpose binary search routine that can be called using the & vector.

MICROTM



FOR YOUR APPLE II

Industry standard products at super saver discount prices



PARALLEL PRINTERS NEC 8023 or C-ITOH 8510

(Virtually identical) Specifications: • 100 CPS dot matrix printer • 80 column print—136 characters per line • Tractor/friction feed • 7 different print fonts included • 2K printer buffer • Proportional spacing • Bit image graphics and graphic symbols.

NEC 8023 or C-ITOH\$495
NEC 8023 or C-ITOH 8510 with Parallel Interface and Cable\$550
EPSON 100 with Parallel Interface and Cable\$749

Z-80 CARD FOR YOUR APPLE MICROSOFT SOFTCARD

With CP/M* and MBASIC.

(List: \$399)\$289



Best Buy!!! ADVANCED LOGIC SYSTEM Z-CARD With C-PM*

Has everything the Softcard has except MBASIC. Works with Microsoft's disks too.

(List: \$269)Special at \$195



ALS SYNERGIZER

CP/M* operating package with an 80 column video board, CP/M* interface, and 16K memory expansion for Apple II. Permits use of the full range of CP/M* software on Apple II. Includes SuperCALC.

(List: \$749)\$549



U-Z-80 PROCESSOR BOARD (From Europe)

Software compatible with Softcard and ALS Software

.....\$149

MICROSOFT + PREMIUM SYSTEM

Includes Videx Videoterm, Softswitch, Microsoft and Softcard, Microsoft and Z-80 Card, and Osborn CP/M* Manual

.....\$595



JOYSTICK

Takes the place of two Apple Paddle Controllers.

From BMP Enterprises. Heavy duty industrial construction and cable. Non-self centering. With polarity switches for consistent motion control.

(List: \$59)\$39

MONITORS FOR YOUR APPLE

AMDEK 300G (18MHZ Anti-Glare Screen)\$179
NEC 12" HIRES GREEN\$179
SUPER SPECIAL!	
SPECIAL 12" GREEN MONITOR\$99

SPECIAL AND NEW

5 MEGABYTE HARD DISK

For Apple II. Supplied with controller. Use with CP/M, Apple DOS, & Apple Pascal\$1995

5 1/4" DISK DRIVE

Use with standard Apple II disk controller.\$295

5 1/4" FLOPPY DISKS

With hub rings. Box of 10.

With other purchase\$19.95
Without purchase\$23.00

16K MEMORY EXPANSION MODULE

The preferred 16K RAM Expansion Module from PROMETHEUS. Fully compatible with CP/M* and Apple Pascal*. With full 1-year parts and labor warranty. (List: \$169)\$75

WORD PROCESSING SPECIAL WITH WORDSTAR AND SUPERCALC!

Do professional word processing on your APPLE. All necessary hardware and software included. Complete 80 column video display, enhanced character set, 16K memory board, Z-Card with CP/M* software, Wordstar and word processing software and SuperCALC.

(List: \$1,128)Special at \$695



from Prometheus! ExpandaRAM

The only 128K RAM card that lets you start with 16K, 32K, or 64K of memory now and expand to the full 128K later. Fully compatible with Apple Pascal, CP/M*, and Visacalc. No Apple modification required. Memory management system included with all ExpandaRAMs. Disk emulators included with 64K and 128K versions.

MEM-32 Two rows of 16K RAMS make a 32K RAM Card\$209
MEM-64 One row of 64K RAM. With DOS 3.3 disk emulator\$299
MEM-128 Two rows of 64K RAMS installed make a 128K Card. With DOS 3.3 disk emulator\$399
MEM-RKT 64K RAM Add-On-Kits— 64K Dynamic RAMS. Each\$125
VISICALC Expansion Program for MEM-128\$75
MEM-PSL Pascal disk emulator for MEM-128\$45

MODEMS FOR YOUR APPLE II

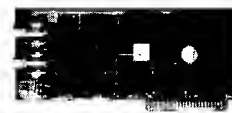
HAYES Smartmodem\$229
MICROMODEM II\$279



VERSAcad FROM PROMETHEUS

Four cards on one! With true simultaneous operation. Includes: (1) Serial Input/Output Interface, (2) Parallel Output Interface, (3) Precision Clock/Calendar, and (4) BSR Control. All on one card. Fully compatible with CP/M* and Apple Pascal*.

(List: \$249)\$169



80 COLUMN VIDEO DISPLAYS FOR APPLE II SMARTERM

(Not to be confused with SUPRTERM)

Software switching from 80 to 40 and 40 to 80 characters. 9 new characters not found on the Apple keyboard. Fully compatible with CP/M* and Apple PASCAL*. With lowest power consumption of only 2.5 watts.

(List: \$345)\$225

SMARTERM EXPANDED CHARACTER SET

7" x 11" matrix with true decenders. Add to above\$40

Best Buy! Combination SMARTERM and EXPANDED CHARACTER SET

Special at\$260
VIDEX, VIDEOTERM\$249
VIDEX ENHANCER II\$119



CENTRONICS COMPATIBLE PARALLEL INTERFACE

From PROMETHEUS. For use with Epson, NEC, C-ITOH, and other printers. Fully compatible with CP/M* and Apple Pascal*.

PRT-1, Only\$69

GRAPHITTI CARD

Prints HIRSE page 1 or 2 from onboard firmware. Features: True 1:1 aspect ratio, prints emphasized mode, reverse mode, rotates 90 degrees ... plus more. Compare all this with the Grappler. We think you'll agree that this is the best graphics card on the market. Specify for use with EPSON, NEC-8023, C-ITOH Prowriter, or Okidata.

(List: \$125)\$89

SOFTWARE

WORDSTARSpecial at \$195
SPELLSTAR\$125
SUPERCALC\$175
D BASE II\$525
VISICALC\$149
DB MASTER\$189

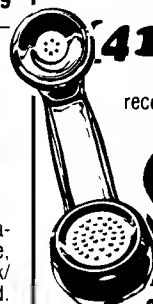
All equipment shipped factory fresh. Manufacturers' warranties included. Please add \$3.00 per product for shipping and handling. California, add 6% tax, BART Counties: 6 1/2%.

All items are normally in stock

**Phone for Quick
Shipment!**

(415) 490-3420

... And we'll be here to help after you receive your order. Feel free to call the SGC Technical Staff for assistance.



SGC

The mail order specialists

342 Quartz Circle, Livermore, CA 94550

MICROTM

THE 6502/6809 JOURNAL

STAFF

President/Editor-in-Chief
ROBERT M. TRIPP

Publisher
MARY GRACE SMITH

Editorial Staff
PHIL DALEY — Technical editor
JOHN HEDDERMAN — Jr. programmer
MARJORIE MORSE — Editor
JOAN WITHAM — Editorial assistant
LOREN WRIGHT — Technical editor

Graphics Department
HELEN BETZ — Director
PAULA M. KRAMER — Production mgr.
EMMALYN H. BENTLEY — Typesetter

Sales and Marketing
CATHI BLAND — Advertising mgr.
CAROL A. STARK — Circulation mgr.
LINDA HENS DILL — Dealer sales
MAUREEN DUBE — Promotion

Accounting Department
DONNA M. TRIPP — Comptroller
KAY COLLINS — Bookkeeper
EILEEN ENOS — Bookkeeper

Contributing Editors
CORNELIS BONGERS
DAVE MALMBERG
JOHN STEINER
JIM STRASMA
PAUL SWANSON
RICHARD VILE

Subscription/Dealer inquiries
(617) 256-5515

DEPARTMENTS

3 December Highlights
7 Editorial
9 Letterbox
30 CoCo Bits
32 From Here to ATARI
35 MICRO News
54 PET Vet
65 APPLE Slices
91 Updates/Microbes
97 Short Subjects
99 New Publications
100 Reviews in Brief
103 Software Catalog
107 Hardware Catalog
108 6809 Bibliography
109 Data Sheet
111 Advertiser's Index
112 Next Month in MICRO

COMMODORE FEATURE

- 37 It's All Relative — CBM Disk Techniques, Part 1**.....James Strasma
Get the most from CBM's powerful disk operating system
- 42 Squeeze for PET Programs**.....Hans Hoogstraat
Squeeze out imbedded blanks, line separators, and comments
- 47 BASIC Line Delete for PET/CBM and VIC**.....Thomas Henry
A machine-language program to delete blocks of BASIC lines
- 52 SOUP: A CBM Machine-Language Compare Program**.....Henry Troup and James Strasma
A compare program for machine-language program files
- 59 Microcomputers in a College Teaching Laboratory, Part 2**.....Richard Heist, Thor Olsen, and Howard Saltsburg
Analog transducers in a digital world

BASIC AIDS

- 13 APPLE ILISZT for Integer BASIC Programs**.....Leonard Anderson
Print your program in a clear, structured format and detect embedded binary code
- 19 BASIC Macro Function for Cursor Control on the OSI**.....Kerry Lourash
Insert statements with just two keys
- 22 ATARI Character Graphics from BASIC, Part 3**.....Paul Swanson
Add to ATARI's vertical blank interrupt routines
- 26 APPLESOFT GOTO/GOSUB Checking Routine**...Peter J.G. Meyer
Verify all GOTO and GOSUB references in your program

HARDWARE

- 69 Adding Voice to a Computer**.....Michael E. Valdez
A low-cost procedure for sampling and reproducing voice
- 74 Enhanced Video for OSI C1P**.....David Cantrell and Terry Terrence
Add five chips — and several features
- 77 Home Control Interface for C1P**.....John Krout
Add your own ultrasonic control
- 82 ATARI Meets the BSR X-10**.....David A. Hayes
Use ATARI's controller ports
- 85 68000 Logic Instructions**.....Joe Hootman
Our discussion of the 68000 instruction set continues
- 88 Programmable Character Generator for OSI**.....Colin Macauley
Design your own character set
- 93 Utilizing the 6502's Undefined Operation Codes**.....Curtis Nelson, Richard Villarreal, and Rod Heisler
Hardware to use these op codes for new pseudo-instructions

Lyc0 Computer Marketing & Consultants

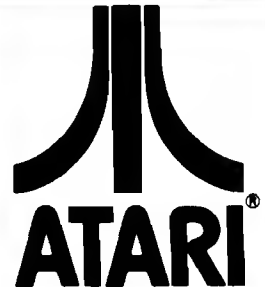
TO ORDER
CALL US

TOLL FREE 800-233-8760
In PA 1-717-398-4079

December
ATARI
SPECIALS

810 Disk Drive ... \$ 429.00
32K RAM \$ 79.00
400 32K RAM ... \$349.00

800 48K... \$609.00



A Warner Communications Company

PERCOM : In Stock

Single Drive CALL
Dual Drive CALL
(Read all Atari Disks)

PRINTERS : In Stock

Epson Mx 80 \$449.00
Epson Mx 80 FT III \$499.00
Okidata 82A \$479.00
Okidata 83A \$719.00
Okidata 84 \$1089.00
Citoh CALL
Prowriter I \$499.00
Prowriter II CALL
SMITH CORONA TP-1 \$625.00
NEC CALL
(Interfacing Available)

JOYSTICKS : In Stock

Atari CX-40 \$18.00
LeStick \$34.00
Wico Command Control \$24.00
WICO RED BALL \$27.95
STICK STAND \$ 8.75

Computer Covers

800 \$8.99
400 \$8.99
810 \$8.99

DISKETTES : In Stock

Maxell MD1 (10) \$34.00
Maxell MD2 (10) \$44.00
Elephant (10) \$21.00

THIRD PARTY SOFTWARE ATARI PROGRAM EXCHANGE

Eastern Front 1941 \$25.50
Avalanche \$15.50
Outlaw/Howitzer \$15.50
Dog Daze \$15.50
Wizard of War \$31.00
Gorf \$31.00
Frogger \$28.00

BUSINESS SOFTWARE : In Stock

Atari Word Processing \$109.00
Letter Perfect \$129.00
Test Wizzard \$ 89.00
Datasm/85 \$125.00
Interlisp \$125.00

Monkey Wrench \$ 42.00
Utility Disk \$ 36.50
Ultimate Renumbr \$ 15.50

ATARI HARDWARE

410 Cassette Recorder \$75.00
825 Printer \$585.00
830 Phone Modem \$149.00
850 Interface \$184.00

PACKAGES

CX481 Entertainer \$69.00
CX482 Educator \$125.00
CX483 Programmer \$49.00
CX494 Communicator \$325.00

SOFTWARE

CXL4012 MISSILE COMMAND \$28.75
CXL4013 ASTEROID \$28.75
CXL4020 CENTIPEDE \$32.75
CXL4022 PACMAN \$32.75
CXL4011 STAR RAIDER \$34.75
CXL4004 BASKETBALL \$28.75
CXL4006 SUPER BREAKOUT \$28.75
CXL4008 SPACE INVADER \$28.75
CX8130 CAVERNS OF MARS \$31.75
CX4106 HANGMAN \$12.75
CX4102 KINGDOM \$12.75
CX4112 STATES & CAPITALS \$12.75
CX4114 EUROPEAN COUNTRIES \$12.75
CX4109 GRAPHIT \$18.75
CX4121 ENERGY CZAR \$12.75
CX4123 SCRAM \$19.75
CX4101 PROGRAMMING I \$19.75
CX4108 PROGRAMMING II \$22.75
CX4117 PROGRAMMING III \$22.75
CXL4015 TELELINK \$21.75
CX4119 FRENCH \$39.75
CX4118 GERMAN \$39.75
CX4120 SPANISH \$39.75
CX4120 SPANISH \$39.75
CXL4007 MUSIC COMPOSER \$33.75
CXL4002 ATARI BASIC \$45.75
CX8128 MICROSOFT BASIC \$65.75
CXL4003 ASSEMBLER EDITOR \$45.75
CX8128 MACROASSEMBLER \$89.75
CXL4018 PILOT HOME \$85.75
CX405 PILOT EDUCATOR \$99.75
CX415 HOME FILING MANAGER \$41.75
CX414 BOOKKEEPER \$119.75

NEW RELEASES

CHOP LIFTER \$27.75
APPLE PANIC \$23.75
PREPIE \$19.95

THIRD PARTY SOFTWARE

for atari 800 or 400
K-BYTE

KRAZY SHOOTOUT \$35.00
K-DOS \$65.00
K-STAR PATROL \$37.75
K-RAZY ANTICS \$37.75
K-RAZY KRITTERS \$37.75
Q-BALL JOYSTICK KIT \$6.75

AUTOMATED SIMULATIONS

Star Warrior \$28.00
Crush, Crumble & Chomp \$23.00

WE CARRY MANY OTHER THIRD PARTY PRODUCTS
YOU CAN CALL FOR PRICES ON AND ASK FOR
YOUR FREE ATARI PRODUCT CATALOG.



VIC-20 \$189.00

VIC1530 DATASSETTE \$67.00
VIC1540 DISK DRIVE \$499.00
VIC1515 PRINTER \$355.00
VIC1210 3K RAM \$35.00
VIC1110 8K RAM \$52.00
VIC1211A SUPER EXPANDER \$53.00

VIC-20 SOFTWARE

VIC1212 PROGRAMMER AID \$45.00
VIC1213 VICMON \$45.00
VIC1906 SUPER ALIEN \$23.00
VIC1914 ADVENTURE
LAND ADVENTURE \$35.00
VIC1915 PRIVATE COVE
ADVENTURE \$35.00
VIC1916 MISSION IMPOSSIBLE \$35.00
VIC1917 THE COUNT ADVENTURE \$35.00
VIC1919 SARGON II CHESS \$35.00

THIRD PARTY SOFTWARE

ALIEN BLITZ \$21.00
Omega Race \$35.00
Gorf \$32.00
16K RAM/ROM \$99.00
AMOK \$21.00
SUPER HANGMAN \$16.00
SPIDERS OF MARS \$45.00



POLICY



In-Stock items shipped within 24 hours of order.
Personal checks require four weeks clearance
before shipping. PA residents add sales tax.
All products subject to availability and price
change. Add 4 % for Mastercard and Visa.

TO ORDER
CALL TOLL FREE
800-233-8760
In PA 1-717-398-4079
or send order to
Lyc0 Computer
P.O. Box 5088
Jersey Shore, PA 17740

About the Cover

100 Fröhliche Weihnachten,
200 Joyeux Noël
300 Felices Pascuas
400 Buon Natale
500 Happy Holidays

This month MICRO is taking a holiday from presenting a graphic with a computer theme on our cover. Instead, we want to offer our warmest greetings — in five languages. The colorful lights in the picture belong to the city of Frankfurt, Germany and symbolize the festive glow of the holiday season. Froliche Weihnachten!

Cover photo by Phil Daley

MICRO is published monthly by:
MICRO INK, Chelmsford, MA 01824
Second Class postage paid at:
Chelmsford, MA 01824 and additional
mailing offices
USPS Publication Number: 483470
ISSN: 0271-9002

Send subscriptions, change of address, USPS
Form 3579, requests for back issues and all
other fulfillment questions to

MICRO INK
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824
or call
617/256-5515
Telex: 955329 TLX SRVC
800-227-1617

Subscription Rates	Per Year
U.S.	\$24.00
	2 yr. / \$42.00
Foreign surface mail	\$27.00
Air mail:	
Europe	\$42.00
Mexico, Central America, Middle East, North Africa, Central Africa	\$48.00
South America, South Africa, Far East, Australasia, New Zealand	\$72.00

Copyright© 1982 by MICRO INK
All Rights Reserved

MICRO™

Editorial

Getting to Know You

"It's more useful than my Swiss army knife." Now that's what we like to hear about MICRO and that's what one of you said in response to our reader survey. But we did the survey for more than a pat on the back.

We did the survey to find out just as much as we can about who you are and what kind of information, both in editorial content and advertising, you need and want.

We discovered that you are an extremely well-educated, affluent, gainfully employed bunch of people with a great deal of technical computer knowledge at your command — and you want more.

33% of you have advanced degrees
70% have incomes over \$25,000
60% are programmer/analysts, engineers, or technicians, and
90% of you have intermediate to advanced knowledge of software and 80% of hardware.

No wonder only 6% of our readers consider MICRO too technical. Your biggest beef? Not enough information on your own system — whatever that may be. Too much Apple, not enough Apple, not enough Atari, not enough OSI. Now we know that that is going to be something of a problem in a publication that covers more than one system, or more than one chip, but we think it's important to cross-fertilize, to generalize, to bring you knowledge and information that is transferable. Our goal is to make at least half of the magazine non-system specific, while dividing the other half in much the way our readers are divided — about half Apple and the other half heavily weighted toward OSI, Commodore, Atari, and 6809 systems. Interest in the 6809 and 68000 remains high, especially among users who are adding boards and processors to 6502 machines.

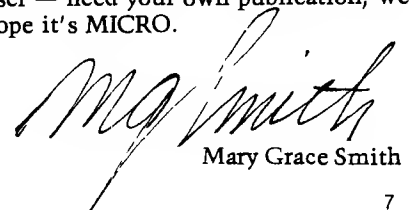
A great many of you (62%) use more than one kind of system and 46% have systems both at home and at work; nearly all of you plan to spend money adding more equipment during the coming year. We trust that the reviews, hardware and software catalogs, and advertisements are helping you make those purchases.

There is a great proliferation of system-specific publications and more and more information for the beginning computer user. We are trying not to

clutter up the magazine with information you already have — you've learned a lot over the last few years and we want to help you build on that knowledge. You've matured, the market has matured, and MICRO is growing along with you. The system-specific magazines are a great place to get hints, corrections, fixes, and details about your own equipment — the kind of material it made sense for us to publish back in 1977 when no one else covered the 6502. But now that manufacturers are doing a better job of providing documentation and there are lots of publications for beginners, we want to concentrate on more advanced issues that cut across machine and processor lines, that keep you abreast of new developments and stretch your knowledge into new areas.

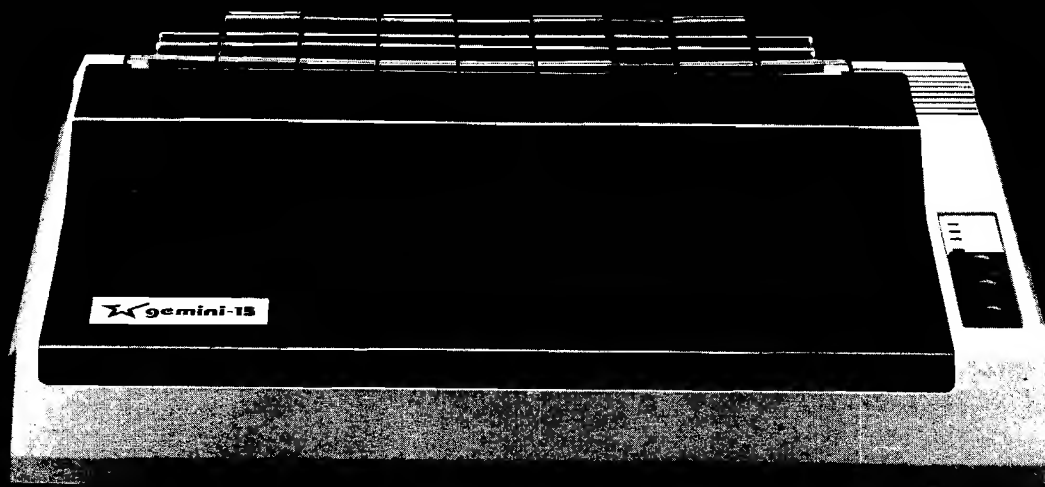
MICRO's editorial schedule for the next year reflects that concern. This is the last system-specific feature we'll be running. Upcoming issues will feature various kinds of peripherals, languages, operating systems, communications. With your strong engineering background you'll want to know what new processors are being developed and how they can be used even before they're available in complete systems. There are new programming languages being developed — we will look at what they are, which ones are worth pursuing for what purposes, etc. We will provide information in the form of data sheets and information sheets on a variety of products and issues. And most interesting of all we will explore new modes of computer use: e.g., networks, communications, automated offices, and industrial control systems.

We think that advanced computer expertise is best imparted in a journal that doesn't limit itself to one system or one chip or one operating system. After all, the whole industry is moving toward compatibility and we think that is a step in the right direction. In light of that fact, and as a result of all we've learned about you and your interests from the survey, as of next month (i.e., with the January 1983 issue), we will change MICRO's subtitle to "*Advancing Computer Knowledge*." We are in no way abandoning the 6502 or the 6809 or any of the specific systems we've been covering. We are, instead, making a statement about your technical expertise, your maturity and the industry's, and our desire to move toward ever increasing compatibility and wider proliferation of advanced information and knowledge. You — the sophisticated user — need your own publication; we hope it's MICRO.


Mary Grace Smith



GEMINI— FOR PRINTER VALUE THAT'S OUT OF THIS WORLD



Over thirty years of down-to-earth experience as a precision parts manufacturer has enabled Star to produce the Gemini series of dot matrix printers—a stellar combination of printer quality, flexibility, and reliability. And for a list price of nearly 25% less than the best selling competitor.

The Gemini 10 has a 10" carriage and the Gemini 15 a 15½" carriage. Plus, the Gemini 15 has the added capability of a bottom paper feed. In both models, Gemini quality means a print speed of 100 cps, high-resolution bit image and block graphics, and extra fast forms feed.

Gemini's flexibility is embodied in its diverse specialized printing capabilities such as super/sub script, underlining, back-spacing, double strike mode and emphasized print mode. Another extraordinary standard

feature is a 2.3K buffer. An additional 4K is optional. That's twice the memory of leading, comparable printers. And Gemini is compatible with most software packages that support the leading printers.

Gemini reliability is more than just a promise. It's as concrete as a 180 day warranty (90 days for ribbon and print head), a mean time between failure rate of 5 million lines, a print head life of over 100 million characters, and a 100% duty cycle that allows the Gemini to print continuously. Plus, prompt, nationwide service is readily available.

So if you're looking for an incredibly high-quality, low-cost printer that's out of this world, look to the manufacturer with its feet on the ground—Star and the Gemini 10, Gemini 15 dot matrix printers.

star
MICRONICS • INC

MAKING A NAME FOR OURSELVES

1120 Empire Central Place, Suite 216, Dallas, TX 75247
For more information, please call Bob Hazzard, Vice President, at (214) 631-8560.

MICROTM

Letterbox

Back and FORTH

Dear Editor:

I was quite pleased with the two articles on FORTH in the June issue of MICRO. Regarding the benchmark comparisons of BASIC, FORTH, and RPL (page 63), I would have to say that Mr. Stryker is apparently somewhat biased in his viewpoint, since he is the father of RPL. What he appears to have done is take perfectly readable FORTH and translate it into hieroglyphics. Surely, the FORTH word DUP is more meaningful as a stack operator than "#", and who would ever guess what ";", ".", and "%" have to do with anything? Single-character words are very useful for lazy typists, but they do tend to produce "write-only" code for those who need to determine what a program is doing.

Every FORTH implementation I have ever seen has a machine-language primitive to handle block moves on a character basis. Why do we go through the gyrations of listing 1B when the word CMOVE would do just as well (actually better!)? Even without using CMOVE, the word BLKM would execute faster and with fewer FORTH words if it were written:

```
: BLKM OVER + SWAP
  DO DUP C@ | C! 1 +
  LOOP DROP ;
```

This word expects a slightly different order of things to be on the stack than originally specified: FROM TO and COUNT (634 826 150 using his numbers). This is the same order that CMOVE would expect them also. I am sure that this arrangement would be of benefit for RPL as well.

Regarding the SHUFFLER benchmark; first of all, it appears there is a typographical error of omission in line 8 of listing 2B, since the word MOD referred to in the text is not there. Even so, however, the way the routine was implemented can do nothing but slow it down.

Finally, regarding the Falling-Tone benchmark, I certainly feel the author's

comments on page 68 regarding how hard it was to come up with a FORTH implementation, show a decided lack of understanding of structured programming! Listing 3A shows the same lack of structure that can be no way blamed on BASIC itself. After analyzing what the program is supposed to do, the following structured code would have been much clearer:

```
1010 DC = 20:FOR Z = 20 TO 255
1020 DC = DC - Z
1030 IF DC >= 0 THEN 1020
1040 POKE 59464,Z
1050 DC = DC + 256
1060 NEXT
1070 POKE 59467,0:POKE
      59466,0:RETURN
```

The same code written in FORTH looks like this:

```
: TONE 0 59464 C! 16 59467 C!
170 59466 C! 20 256 OVER DO
      BEGIN I - DUP 0 <
      UNTIL
      I 59464 C! 256 +
      LOOP DROP 0 59466 ! ;
```

Notice that we use 0 59466 ! to reset both 59466 and 59467 to zero, since FORTH inherently works with 16-bit numbers and uses 8-bit numbers only occasionally. I would probably do the same thing at the beginning of TONE to set up 59466 and 59467 initially, assuming this is a PIA register address of some sort. At any rate, the structure is there and can also be used in the RPL version, I'm sure.

Edward B. Beach
5112 Williamsburg Blvd.
Arlington, VA 22207

Dear Editor:

In "BASIC, FORTH, and RPL" (MICRO 49:63), three different computer languages are compared in terms of speed and memory economy using three benchmark programs. However, within the text of the article there were some comments made about FORTH

by the author, Timothy Stryker, which require rebuttal.

Mr. Stryker states that program modules in RPL do not execute directly but rather place their address on the stack where a second call operator (&) actually executes this address. As correctly noted, this is in contrast to FORTH where the defined word directly executes; it does not need a second execute operator. This allows *all* FORTH definitions to be treated as syntactically equal. Programmers may freely mix FORTH language words with their own new definitions — indeed, there is no difference in the internal dictionary structure between these two parts.

On the other hand, RPL forces us to use (&) for execution of all new words while pre-existing ones are immune to this rule and execute directly, creating an inconsistent syntax. That this is memory efficient is doubtful. The higher level definitions of any non-trivial application program can consist of a large proportion of user-defined operators, each one of which would require the addition of this execute operator in RPL. This probably consumes some memory in the compiled form and it certainly and unnecessarily clutters up the source code. With FORTH, the address of any definition can be placed on the stack with an additional operator when it is desired, although this function is seldom needed.

It is true that FORTH handles symbols differently depending on whether they are variables, constants, or executing subroutine names. This is part of the beauty of the language, not a weakness. Each type of symbol has a different function. Subroutine names execute, constants leave their value on the stack, and variables leave their address so we can suffix them with load or store operators. Nothing could be simpler or more efficient: uniformity of function by means of inconsistent internal operation. RPL reverses this, giving us consistent internal operation while forsaking clarity of function at the programmer's level. This forces us

Letterbox (continued)

to be even more aware of what each definition does — something I would prefer to be left up to my compiler.

As Mr. Stryker correctly states, the FORTH string literal print word (.'') and the numeric print words never leave their output string on the stack. This is seldom needed and would possibly slow down the system. Besides, the stack may not be large enough to safely handle this, since on the 6502 the FORTH stack is placed in page zero (shared with a few other FORTH locations and probably some used by the host computer for disk or terminal I/O). If we need to alter the string in numeric conversion and printing, FORTH has some primitives available for inserting additional characters in the string. With a minor effort we can add print using to an application program or make it a permanent part of the FORTH we use each day. Other than the string literal defining word (.''), there are no other string operators defined in the FORTH standards, but these are not difficult to add to such an easily extensible language.

Some additional points: The modulo primitive in the fig-FORTH 6502 model takes 1.2 milliseconds to execute. No random-number generator is defined by the Group, so the poor speed of this word in Mr. Stryker's unnamed FORTH version was not optimized for speed by whomever wrote it.

Language experimentation and comparison is certainly needed to fuel the evolutionary process of computer technology. But it should best be done with the full understanding of each language involved.

Raymond Weisling
Jalan Citropuran No. 23
Solo, Jawa Tengah
Indonesia

Dear MICRO:

Thanks very much for the chance to respond to Mr. Beach and Mr. Weisling in regard to their letters concerning my recent article.

First of all, I take exception to the contention in both of these letters that I unjustly biased the benchmarks and the conclusions drawn therefrom in

favor of RPL. In fact, precisely *because* I knew that this objection might be raised, I bent over backward to give the benefit of every doubt to FORTH. This may not be immediately apparent in the article because I did not make a point of saying so, but, for example, wherever my measured execution times varied slightly from one run to the next, I uniformly presented FORTH's fastest time, and RPL's slowest; for another, I specifically excluded from consideration any benchmarks involving manipulation of character strings, stack-resident arrays, finite-state automata, and other operations that RPL handles much more naturally than FORTH. Further evidence of this concern will become apparent below.

First I'll address Mr. Beach and his comments on the use of single-character operator-tokens. I do agree that RPL source must look like hieroglyphics to a person versed in FORTH — but perhaps you remember what FORTH (or any computer language) looked like before you became fluent in it. Experienced RPL users have as little difficulty reading RPL source as you do

OMNIFILE

- full featured file manager and report generator for home, business, school, or scientific applications
- user definable file structures
- powerful search and edit, including global change and delete
- built-in statistical analysis
- flexible tabular report and mailing label capabilities, complete with search/sort capabilities on any field

OMNITREND

- powerful multiple regression trend analysis tool for business or technical data
- sophisticated least squares fitting algorithm — faster and more accurate than usual techniques
- includes descriptive statistics and bivariate analysis
- built-in data management and file editing
- extensive built-in hi-res graphics to aid in data analysis

- Professional quality • Unlocked diskettes • Exact
- Apple II Requires Applesoft in ROM, 48K RAM, 128K
- Apple III Requires Business Basic and Minimum 128K

Distributed by

Educational Computing Systems

106 Fairbanks Rd., Oak Ridge, TN 37830 • 615/526-1111

THE OMNIWARE PROFESSIONAL SERIES

OMNIPLOT

Plot for business, scientific, and technical data. Combined bar and line charts (including stacked line and bar charts). Plots, but all plot parameters to allow full control.

Linear transformations on data. Multiple graph labels.

OMNIPACK

Interchangeable with the file, trend, and graph programs (interchangeable).



NEW OMNICOMP

- powerful data manipulation and numerical analysis system
- performs polynomial curve fitting, numerical interpolation, numerical integration, numerical differentiation and statistical calculations using entire data file or selected subsets
- extensive built-in hi-res graphics
- mathematical data transformations, plus averaging, smoothing, and lag/lead
- data files interchangeable with OMNIPACK programs

Apple II	Apple III
\$74.95	\$74.95
\$74.95	\$74.95
\$64.95	\$64.95
\$129.95	\$169.95
\$79.95	\$99.95

Shipping in TN add 4 1/2% tax

OMNI is a trademark of ENDAC, Inc.

Letterbox (continued)

reading FORTH. The advantages of single-character operator-tokens are three: 1. as you acknowledge, they cut down on typing time; 2. they cut down on the physical size of the source, so that more source can be fit into memory at once when undertaking nontrivial applications; and 3. they speed up compilation by cutting down on the operator-token search time.

Thank you for pointing out a better method of doing block moves in both FORTH and RPL. In writing the benchmarks, I was primarily concerned about making sure that the FORTH and RPL versions were as close to identical in approach as possible, so I missed seeing that the block move could be done more efficiently in the way you suggest. You may be interested to know, though, that the FORTH source you show for this routine yields an execution jiffy-count of 717, considerably in excess of the 591 given for FORTH in the article. The reason? Your use of the composite "1+" operator in the innermost loop. When the sequence "1+" is substituted for this, the execution time falls to 584 jiffies. Spaces, as you note in your letter *are* important in FORTH — one might even say, alarmingly so. They make no difference in RPL. Unfortunately, the use of even the sped-up form of your block-move algorithm does not change the standings. FORTH requires 84 program bytes to do it in 584 jiffies, whereas the following RPL equivalent:

```
BLKM: ; + 1 - % FOR # PEEK FN  
POKE 1 + NEXT . RETURN
```

requires only 52 bytes to do it in 508, a "merit ratio" of 1.85 to 1.

Now, there seems to be some confusion in your letter regarding various aspects of the SHUFFLER benchmark. To begin with, there are no typos anywhere in the article. The MOD routine is, as stated, internal to the RND routine I used. This RND routine, modeled after that available under MMSFORTH, expects an integer passed to it on the stack, and returns a random number in the range from 0 up to that integer minus 1 — hence, the MOD.

Moving on to your comments regarding the third benchmark: you are right. There was no need for me to introduce unstructured code in this case.

The new FORTH TONE routine you exhibit takes only 3465 jiffies, and requires only 130 bytes of program space. The corresponding RPL routine is:

```
TONE: 0 59464 POKE 16 59467 POKE  
170 59466 POKE 20 256 ; FOR  
LOOP: FN - # 0 < IF  
FN 59464 POKE 256 +  
THEN LOOP GOTO END  
NEXT . 0 59466 ! RETURN
```

which requires 83 bytes of storage and executes in 3338 jiffies. The resulting merit ratio of 1.62 to 1 represents a considerable improvement. You were right, incidentally, not to condense the leading POKEs of 59467 and 59466 into a single store — the order of the POKEs into those 6522 VIA registers makes a big difference.

On to Mr. Weisling's letter. Programmers who are bothered by the necessity of suffixing their subroutine references with an ampersand in RPL are free to eliminate the space separating the two and thereby regard the composite "SUBRNAME&" as just a one-keystroke-longer method of invoking the routine. You doubt that this is memory efficient. Please find out for certain by way of the following procedure: take any nontrivial FORTH application program to which you have access and count up the number of occurrences of (A) invocations of the thirty or forty real low-level FORTH "primitives" such as DUP, "=", IF, DO, "@", and things of that nature (including ";" but not including ":"); (B) references to literal numeric quantities, whether CONSTANTS or not, it does not matter, which fall in the range from 0 to 63; (C) references to literal numeric quantities greater than 63 but less than 32768, plus all references to VARIABLES, CVARIABLEs, and what-not; (D) all references to literal numeric quantities not covered under B or C; and (E) all routine-invocations (other than ":") not covered under A. Be sure, if you count a routine-invocation under E, that you also consider the body of that routine part of the program source. Now form the sum $A + B + 2 \cdot C + 3 \cdot D + 3 \cdot E$. This is a rough approximation of the number of object program bytes that would be required, were the program translated, absolutely mechanically from FORTH into RPL. Multiply this by about 0.8 to arrive at the memory size of the

equivalent program, had it been designed in RPL to begin with.

Next, a discussion on symbol handling. The fact that RPL is more efficient has been demonstrated already. That it is simpler may be difficult to appreciate second-hand like this, but RPL "gives us consistent internal operation" without forsaking "clarity of function at the programmer's level." The question of how aware the programmer needs to be as to "what each definition does" has nothing to do with it.

The ability to manipulate character strings conveniently is fundamental to most user-oriented software development. Indeed, your remark about the size and location of the FORTH stack points up the fact that this is one area in which FORTH's extensibility does it little good. RPL locates both stacks in page one: the parameter stack is the hardware stack, and the return stack is an indexed sort of affair down below it. Stack-resident strings up to 60 characters long or so can be manipulated freely without fear of crashing the machine — and execution is brought to a controlled halt if the 64-word stack entry limit is exceeded.

And on your last point: under my version of FORTH, a public-domain version identifying itself simply as "fig-FORTH 1.0" (which, however, includes such exotic facilities as double-precision and floating-point math, IEEE-488 I/O, etc.), the following routine, as timed with an actual watch, takes 2 minutes and 40 seconds to execute:

```
: TEST 30000 0 DO 6543 52 MOD DROP  
LOOP ;
```

When the MOD is replaced with another DROP, it takes 14 seconds. I leave you to draw your own conclusions.

Timothy Stryker
Samurai Software
P.O. Box 2902
Pompano Beach, FL 33062

MICRO™

Your opinions, comments, and criticisms can be aired in MICRO too. Send mail to Letterbox, MICRO, P.O. Box 6502, Chelmsford, MA 01824.



**CHRISTMAS SEASON
SPECIALS!**
Let ARK COMPUTING Make This Your
Best Christmas Ever!

Super Fan II by R.H. Electronics **59.95/79.95**

Applicard, a high performance Z-80 card
with 64K Ram, complete with CP/M
4 mhz **324.95/445.00**
6 mhz **395.00/595.00**

Microsoft Z-80 card with CP/M and
Microsoft Basic
2 mhz **269.95/395.00**

Microtek Parallel Printer Interface complete
with centronic compatible connector
64.95/79.95

Lazer Lower Case +Plus with Character
Set +Plus **49.95/84.90**
Lower Case +Plus alone **39.95/59.95**

Lazer Graphics +Plus **99.95/159.95**
Graphics +Plus and
Lower Case +Plus **134.95/219.90**

Computer Stop 16K Ram Board **69.95/149.95**

Computer Stop Omnivision 80 Column board
129.95/295.00

Videx Video-term with softswitch, inverse
character set and 80 column Visicalc preboot
295.00/450.00

Wizard BPO 16K buffered printer interface
(expandable to 32K) **134.95/179.95**

Wizard 80, 80 column board **195.00/295.00**

Lazer Pascal **29.95/39.95**

Anix 1.0 **34.95/49.95**

Lazer Forth **44.95/59.95**

D Tack 68000 board for the Apple II
with 4K Ram **895.00**

Lazer Model/32 (16032 board for the Apple II)
CALL!

Lisa	59.95/79.95
Lisa Educational Pak	79.95/119.95
Alien Ambush	19.95/29.95
Bandits	19.95/29.95
Cannonball Blitz	24.95/34.95
County Fair	19.95/29.95
Cranston Manor	24.95/34.95
Cyclod	19.95/29.95
David's Midnight Magic	24.95/34.95
Dosource 3.3	24.95/39.95
Dueling Digits	19.95/29.95
Falcons	21.95/29.95
Firebird	21.95/29.95
Foosball	19.95/29.95
Horizon V	25.95/34.95
Genetic Drift	19.95/29.95
Kabul Spy	24.95/34.95
Jelly Fish	19.95/29.95
Lemmings	19.95/29.95
Labyrinth	19.95/29.95
Mouskattack	24.95/34.95
Outpost	19.95/29.95
Red Alert	19.95/29.95
Pig Pen	24.95/34.95
Ruski Duck	25.95/34.95
Minator	24.95/34.95
Track Attack	19.95/29.95
Thief	17.95/29.95
Space Quarks	19.95/29.95
Snack Attack	19.95/29.95
Swash Buckler	24.95/34.95
Gin Rummy	24.95/34.95
The Dictionary	69.95/99.95
General Manager	99.95/149.95
4 Ft. Disk Cable	19.95/29.95
Visicalc	179.95/250.00
Using 6502 Assembly Language Book	14.95/19.95
Kids and The Apple Computer Book	15.95/19.95
Apple Panic	19.95/29.95
Kraft Joystick	49.95/69.95



714 735-2250
P.O. Box 2025
Corona, CA 91720



APPLE ILISZT for Integer BASIC Programs

by Leonard Anderson

ILISZT prints an Integer BASIC program in a clear, structured format with the ability to detect embedded or attached BINARY code.

ILISZT

requires:

Apple II with both
Integer and Applesoft
Disk drive
Printer

The purchase of several disks at the end of 1981 added a number of Integer BASIC programs to my Apple II library. No listings were available and I decided to print all of them.¹ Several had embedded binary code, a condition that caused much "nonsense" display on both screen and printer. "LISZT" was already up and running (MICRO 48:37), so it seemed logical to modify this Applesoft program to format Integer listings. The ILISZT result kept the original format and added the ability to find exact binary code addresses.

ILISZTER is the formatting and printing program, run by EXEC file ILISZT. ILISZTER is Applesoft rather than Integer. While an Integer program might seem better, many Apple II owners possess ROM or RAM cards for language duality and ILISZTER seems more compact in Applesoft due to string-handling capability. Another advantage is that ILISZTER can be re-run without disk operations or loss of Integer source code.

ILISZTER retains the original features such as separation of concatenated statements, indenting, and remark highlighting. Multiple-iterator NEXT statement handling for restoring FOR-NEXT loop indents is an improvement. The added binary code determination and restoration routine is useful for listing certain utilities.²

Since Integer BASIC differs from Applesoft, a brief review of Integer structure will help provide an understanding of ILISZTER.

Integer BASIC Source Code

Figure 1 shows one line number of source code in Integer. The first byte contains the number of bytes per line with the next two bytes having the line number in binary. End-of-line is signified by the end byte having a value of one.

Each entered line is immediately checked for syntax. Line numbers are limited to 32767 but may be modified by utilities. Numeric constants are converted to binary on entry, an advantage for program execution time.

All function words are stored as one-byte "tokens" in the range of zero to 127 decimal. Punctuation, arithmetic, and logical operators are also tokens. Eight tokens are unused and three others are used only with keyboard entries. ASCII characters have the high bit set to use the decimal range

of 128 and 255. Token and character values are opposite that of Applesoft.

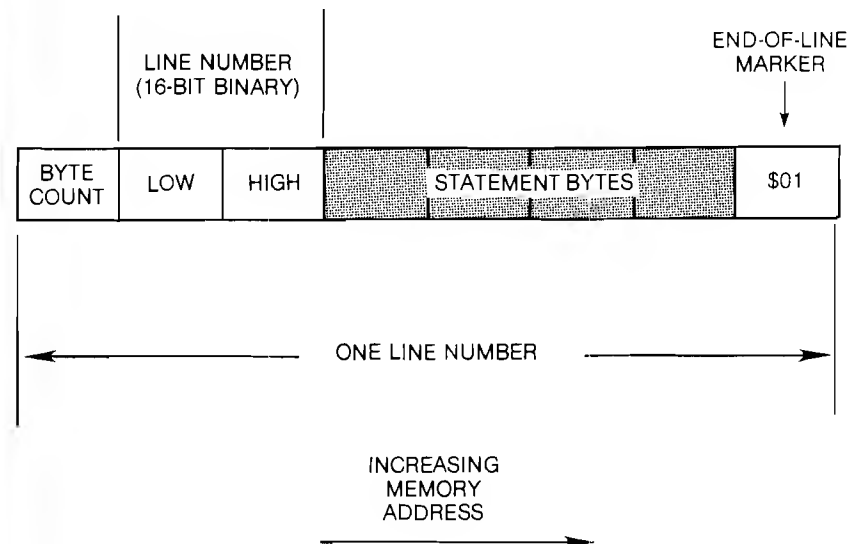
A major difference also exists in handling numeric constants within Integer. Certain functions permit a following numeric constant or variable name. Distinction of a numeric constant is done by making the first byte following an ASCII number (\$B0 to \$B9, not allowed as first letter of a variable) with the next two bytes containing the numeric constant in binary.

Integer BASIC is located just below the highest free memory address. Integer does not need the three-null end of program marker required by Applesoft. Other details may be found in earlier publications.^{3, 4, 5}

An EXEC File for Glue

If an Integer program exists in memory, loading an Applesoft program will not destroy the Integer source code. Loading does change the Integer start-of-program pointer at \$CB, \$CA [203, 202]. Integer end-of-program, or HIMEM at \$4D, \$4C [77, 76] remains unchanged.

Figure 1: Source code structure on one line number in Integer



HIMEM will restore to the end of free memory on re-loading an Integer program; the mechanism is unknown but confirmed through experiments.

EXEC file ILISZT is executed after loading the Integer program to be listed. The first two POKES in ILISZT generator MAKE ILISZT will move the Integer HIMEM pointer into the LOMEM space at \$4B, \$4A (75, 74). LOMEM also restores on Integer reload. The last two POKES move the start-of-program into the space normally used for Integer HIMEM.

Running ILISZTER will automatically switch over to Applesoft without disturbing the new Integer start and end addresses. MAKE ILISZT can be deleted when EXEC text file ILISZT is generated.

Starting ILISZTER

The first line resets Applesoft high memory to prevent string operations from overwriting the Integer source. Token words are initialized at line 91. Since quotes are tokens if not in a remark, the DATA declaration uses an "&" symbol with conversion *via* the IF and CHR\$(34) statement.

A token evaluation array is generated in V at line 96. The V array is used in line parsing to test unused tokens and tokens that may have following numeric constants. Unused tokens (V=2) may be nulls or single spaces; spaces were written just in case the binary-insert routine crashed.

The choice of lower-case characters in token words is up to the user. Mixed-case token words give distinction from normal upper-case variables. Available utilities can edit upper-case source code by adding hexadecimal \$20 to each desired lower-case letter.⁶

Initial display at line 98 is optional but it does indicate proper location and operation. The "DIFFERENT START ADDRESS" prompt allows listing to begin *after* an embedded binary; binary addresses will appear in normal printouts. ILISZTER can be RUN after any RESET or list completion without disturbing Integer source code.

Printer control in lines 107 to 110 should be set to your particular printer and interface. Subroutines at lines 17 and 18 can be changed to other runtime control. Source code control characters are converted to letters before output.

Lines that Parse in the Right

A source code line parse begins at

ILISZTER

```

0 PS = PEEK (77) * 256 + PEEK (76) - 1: HIMEM: PS: GOTO 82
1 REM "GET BYTE" SUBROUTINE *
2 P = P + 1: B = PEEK (P): RETURN
3 REM "BLANK LINE PRINT" SUBROUTINE *
4 D = 0: GOSUB 6: PRINT SS: RETURN
5 REM "TEST PAGE SUBROUTINE *
6 LC = LC + 1: IF LC = < LP THEN RETURN : REM NOT A NEW PAGE
7 GOSUB 17: LC = 6: PC = PC + 1: PRINT SS: PRINT BB$, LB$: "<continued>"
8 REM A FORM-FEED FOR TOP OF NEXT PAGE; ALLOWS VARIATION FOR DIFFERENT P
  RINTERS.
9 FOR K = 1 TO 4: PRINT SS: NEXT
10 REM PRINT THE HEADER
11 HS(4) = "Integer Page " + STR$(PC): FOR K = 1 TO 4: E = INT ((LL -
  LEN (HS(K))) / 2) + 1: PRINT M$: LEFT$(BB$, E): HS(K): NEXT : K = FRE
  (0): PRINT SS: IF NOT D THEN RETURN
12 REM PUT LINE NUMBER IN BRACKETS AS A STATEMENT IDENTIFICATION ON NEXT
  PRINT PAGE
13 NS = STR$( VAL (NS)): K = LEN (NS): REM NS IS NOW WITHOUT SPACES; BR
  ACKET NS AND ATTACH TO STATEMENT CHARACTERS
14 CS = RIGHT$( ( LEFT$( LB$, (6 - K)) + CHR$( 91) + NS + CHR$( 93) + S
  $), B) + RIGHT$( CS, ( LEN (CS) - 8)): K = FRE (0): RETURN
15 REM * * * MX-80 STANDARD/ITALICS SUBROUTINES * * *
16 REM "GRAFTRX" Only. Single-character-set printers should DELETE the
  se calls throughout if not used for other print functions.
17 PRINT CHR$( 27)"5";: RETURN : REM ESC-5 IS STANDARD SET
18 GOSUB 17: IF RF THEN PRINT CHR$( 27)"4";: REM ESC-4 IS ITALICS SET
19 RETURN
20 REM HEXADECIMAL CONVERT SUBROUTINE *
21 AS = "": REM ENTER WITH 'L' AS DECIMAL NUMBER, RETURN IN 'A$'
22 FOR K = 1 TO 4: D = INT (L / 16): E = INT ((L - (D * 16)) + 1): L = D:
  AS = MID$( X$, E, 1) + AS: NEXT : REM PREFIX THE "$" HEX NOTATION
23 AS = "$" + AS: K = FRE (0): RETURN
24 REM BEGIN A NEW LINE NUMBER WITH TEST OF NUMBER OF BYTES IN LINE FROM
  FIRST BYTE, THEN CONVERT BINARY LINE NUMBER TO DECIMAL
25 GOSUB 2: IF P = > PE GOTO 123: REM POINTER EQUAL TO OR BEYOND END OF
  INTEGER PROGRAM
26 LA = P: BC = B: IF B > 127 GOTO 114: REM BYTE COUNT TOO LARGE, PROBABLE
  ATTACHED BINARY
27 TN = TN + 1: REM BUMP LINE NUMBERS, THEN MAKE LINE NUMBER STRING
28 GOSUB 2: L = B: GOSUB 2: L = 8 * 256 + L: B = LEN ( STR$( L)): NS = RIGHT$
  (( LEFT$( LB$, (7 - B)) + STR$( L) + " " ), 8)
29 REM BEGIN STATEMENT LINE PARSING WITH FIRST-BYTE DECISION
30 D = 0: GOSUB 2: IF B = 93 AND NOT RF THEN GOSUB 4: GOTO 34: REM SEPA
  RATE REM-GROUPS BY BLANK LINES
31 IF B = 93 AND RF GOTO 34
32 IF RF THEN RF = 0: GOSUB 4
33 REM RE-ENTRY POINT FOR NEXT BYTE IN STATEMENT DECISION
34 IF B < 128 GOTO 39: REM BYTE IS A TOKEN
35 IF B = 255 THEN B = 159: REM RUBOUT ($FF) BECOMES UNDERLINE BETWEEN B
  ARS
36 B = B - 128: IF B < 32 THEN B = B + 64: GS = GS + CHR$( 124) + CHR$( (
  B): B = 124: REM PUT CONTROL CHARACTERS BETWEEN BARS
37 GS = GS + CHR$( B): GOSUB 2: GOTO 34
38 REM TOKENS
39 IF V(B) > 1 THEN GS = "": GOTO 114: REM UNUSED TOKEN, PROBABLE BINARY
  PROGRAM ATTACHED SO GATHERING IS NULLED
40 IF B = 1 OR B = 3 THEN GS = GS + S$: GOTO 57: REM FORCE A NEW PRINT L
  INE ON E-O-L OR A COLON DELIMITER; SPACE ATTACHED TO PREVENT PRINT-L
  INE CRASH
41 IF B = 93 THEN TR = TR + 1: RF = 1: RS = 1: REM A "REM"
42 IF B = 37 AND PEEK (P + 1) = 85 THEN GS = GS + T$(B): CF = 1: GOTO 57
  : REM FORCE A NEW LINE ON "THEN" FOLLOWED BY "FOR", SET CONDITIONAL
  FLAG
43 IF B = 85 THEN FF = 1: REM A "FOR"
44 IF B < > 89 GOTO 51: REM SKIP AROUND A "NEXT"
45 FS = FS - 1: PT = P + 1: IF CF THEN FS = FS - 1: REM DECREMENT "FOR" SP
  ACER ON "IF" FLAG SET, BEGIN SCANNING AHEAD FOR 2 OR MORE ITERATORS
46 BT = PEEK (PT): IF BT = 1 OR BT = 3 GOTO 49: REM NO OTHER ITERATOR
47 IF BT = 90 THEN FS = FS - 1: REM COMMA FOUND, DECREMENT "FOR" SPACER
48 PT = PT + 1: IF PT < (LA + BC) GOTO 46: REM CHECK AGAIN FOR ANOTHER
  COMMA WITHIN LINE
49 IF FS < 0 THEN FS = 0
50 REM GATHER TOKEN THEN TEST FOR A FOLLOWING 3-BYTE NUMBER GROUP
51 GS = GS + T$(B): L = B: GOSUB 2: IF V(L) = 0 GOTO 34: REM NO NUMBER SHD
  ULD FOLLOW
52 IF B < 176 OR B > 185 GOTO 34: REM THE $B0-$B9 FIRST-BYTE NOT THERE S
  O NO NUMBER FOLLOWS. FALL-THROUGH IGNORES FIRST-BYTE AND DOES DECIM
  AL STRING CONVERSION
53 GOSUB 2: L = B: GOSUB 2: L = B * 256 + L: GS = GS + STR$( L): GOSUB 2: GOTO
  34
54 REM ADD EXTRA INDENT EACH SPLIT LINE, LIMITING ON "REM" STATEMENTS
55 TS = TS - 1: SF = 0: RS = RS + 1: IF RS > 2 THEN RS = 2
56 REM FIRST ENTRY TO PRINT-LINE BUILD, GET TOTAL INDENT SPACES PLUS SPL
  IT-POINT LOW LIMIT 'E'
57 TS = TS + 1: K = IM * (FS + RS): E = K + 13: IF K > 0 THEN GS = LEFT$( (
  BB$, K) + GS

```



```

58 REM BUILD TOTAL PRINT-LINE STRING
59 IF NOT D THEN C$ = N$ + G$
60 IF D THEN C$ = L$ + G$
61 REM TEST FOR LONG LINE, SPLIT IF NECESSARY
62 K = LEN (C$) - LL: IF K < 1 GOTO 74: REM NOT A SPLIT LINE
63 G$ = RIGHT$(C$,K):C$ = LEFT$(C$,LL):SF = 1
64 REM BEGIN SPLITTING WITH SEARCH FOR A SPACE
65 D = LL
66 IF MID$(C$,D,1) = $$ GOTO 72
67 D = D - 1: IF D > E GOTO 66
68 D = LL: REM SPLIT NEXT AT ARITHMETIC OPERATOR OR COMMA
69 K = ASC ( MID$(C$,D,1)): IF K = 42 OR K = 43 OR K = 44 OR K = 45 OR
    K = 47 OR K = 124 GOTO 72
70 D = D - 1: IF D > E GOTO 69: REM FALL-THROUGH IS NO SPLIT
71 GOTO 74: REM NEXT LINE IS SPLITTING INSTRUCTION
72 K = LL - D: IF K > 0 THEN G$ = RIGHT$(C$,K) + G$:C$ = LEFT$(C$,D)
73 REM TEST PAGE LINE-COUNT, INSERT SPACES AS REQUIRED, THEN PRINT
74 GOSUB 6:K = LEN (C$): IF SF = 0 OR K < 2 OR RF THEN 77: REM FORGET M
    ARKING UNDERLINING ON "REM'S
75 IF MID$(C$,K,1) = $$ THEN C$ = LEFT$(C$,K-1)) + CHR$(95): REM
    PUT A TRAILING UNDERLINE AT LAST SPACE AS A MARKER FOR THE LEFT-HAND
    STRING
76 IF LEN (G$) > 2 AND LEFT$(G$,1) = $$ THEN G$ = CHR$(95) + RIGHT$(
    G$,( LEN (G$) - 1)): REM PUT A LEADING UNDERLINE AT FIRST SPACE OF
    RIGHT-HAND STRING AS A MARKER
77 GOSUB 17:K = LEN (C$): PRINT M$: LEFT$(C$,8);: GOSUB 18: PRINT RIGHT$(
    C$,K-8));K = FRE (0): IF SF THEN D = 1: GOTO 55: REM PRINT REST
    OF A SPLIT LINE
78 RS = 0: IF FF THEN FS = FS + 1:FF = 0
79 D = 0:SF = 0:G$ = "": IF B = 1 GOTO 25: REM GET ANOTHER LINE NUMBER IF
    E-O-L, ELSE FALL THROUGH AND GET ANOTHER STATEMENT
80 GOSUB 2:D = 1: GOTO 34
81 REM INITIALIZATION OF VARIABLES
82 DIM T$(127),H$(4),V(127)
83 REM INITIAL VARIABLE SETTING HAS AN 80-CHARACTER WIDE PRINT LINE AND
    82-LINE PAGE LENGTH (INCLUDING HEADER, EXCLUDING 'CONTINUED' INDICAT
    OR); CHANGE LL AND LP AS DESIRED FOR OTHER FORMAT SIZE.
84 PE = PEEK (75) * 256 + PEEK (74) - 1:P = PS: REM PS = INTEGER PROGRA
    M START ADDRESS MINUS ONE, PE = INTEGER PROGRAM STOP ADDRESS MINUS O
    NE
85 B = 0:LL = 80:LP = 82:IM = 4:TN = 0:TS = 0:TR = 0:$S = " ":X$ = "01234
    56789ABCDEF":C$ = "":G$ = "":N$ = "":M$ = "":L$ = " ":B$ = "
    "
86 REM 'T$' ARRAY STRING CONSTANTS FOR PRINTING TOKENS
87 DATA " ", "<<", " ", "<", "Load ", "Save ", "Con", "Run", "Del ", " ",
    "New", "Clear", "Auto ", " ", "Man ", "Himem : ", "Lmem : ", "+", "-", "*", "/"
    , " ", " ", "# ", " ", ">" , " ", " ", "<" , " ", "<" , "And ", "Or ", "Mod
    ", "^", " ", " ", " ", "Then "
88 DATA "Then ", " ", " ", " ", " ", " ", " ", " ", " ", " ", "Peek", "Rnd", "Sgn", "Abs
    ", "Pdl", " ", " ", " ", " ", "Not ", " ", " ", " ", "Len(", "Asc(", "Chr$", "Scrn"
    , " ", " ", "$", " ", " ", " ", " ", " ", " ", " ", " ", " ", "Text", "Gr", "Ca
    ll ", "Dim ", "Dim ", "Tab "
89 DATA "End", "Input ", "Input ", "Input ", "For ", " ", "To ", "Step ", "N
    ext ", " ", "Return", " Gosub ", "*", " ", "GoTo ", "If ", "Print ", "Pri
    nt ", "Print", "Poke ", " ", "Color = ", "Plot ", " ", "Lin ", " ", "At ", "
    Vlin ", " ", "At ", "VTab "
90 DATA " ", " ", " ", " ", " ", "List ", " ", "List", "Pop", "NoDsp ", "NoDsp ", "
    NoTrace", "Dsp ", "Dsp ", "Trace", "Pr # ", "In # "
91 FOR K = 0 TO 127: READ T$(K): IF T$(K) = "&" THEN T$(K) = CHR$(34):
    REM ONE WAY TO GET A DOUBLE QUOTE INTO A STRING
92 NEXT
93 REM 'V' ARRAY CONSTANTS FOR TOKEN TESTING
94 DATA 2,0,2,0,0,0,0,1,0,1,0,0,2,2,2,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
    1,2,1,1,1,0,0,0,0,0,1,2,2,1,0,0,0,0,0,2,1,1,1,1,1,0,0,0,0,1,1,0,2,
    1,1,1,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,0,0,1,1,0,1,0
95 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0,2,1,1,0,0,0,0,0,0,0,1,1
96 FOR K = 0 TO 127: READ V(K): NEXT
97 REM SCREEN PROMPTS AND OPERATOR ALTERNATES
98 HOME : TEXT : VTAB 2: HTAB 12: INVERSE : PRINT " ILISZTER ACTIVE " : NORMAL
    : VTAB 4:L = PS + 1: GOSUB 21: PRINT "START OF INTEGER PROGRAM:";A$:
    :L = PE: GOSUB 21: PRINT " END OF INTEGER PROGRAM:";A$: REM OPTION
    AL TO CHECK APPROXIMATE ADDRESS LOCATION
99 PRINT : INPUT "PROGRAM NAME: ";H$(1): INPUT " PROGRAMMER: ";H$(2): INPUT
    " DATE: ";H$(3): REM REQUIRED FOR HEADER ON EACH PAGE
100 PRINT : PRINT "WANT DIFFERENT START ADDRESS?": GET A$: IF A$ < > "
    Y" GOTO 103
101 INPUT " START ADDRESS (HEX): ";A$:D = 1:BT = 0: FOR K = LEN (A$) TO
    1 STEP - 1: FOR E = 1 TO 16: IF MID$(A$,K,1) = MID$(X$,E,1) THEN
    BT = D * (E - 1) + BT
102 NEXT E:D = D * 16: NEXT K: PRINT :P = BT - 1:L = BT: GOSUB 21: PRINT
    " HEX ADDRESS = ";A$: " CHANGE ?": GET A$: IF A$ = "Y" GOTO 101
103 PRINT : PRINT "NO LEFT MARGIN, WANT ONE?": GET A$: IF A$ = "Y" THEN
    INPUT " MARGIN SPACES: ";K: IF K > 0 AND K < 49 THEN M$ = LEFT$(
    B$,K):LL = LL - K
104 REM REMINDER FOR PRINTER SET-UP

```

line 25. Integer does not allow a byte count larger than 127. (The actual number is 255. The 127-byte limit (line 26) is for print-line reconstruction, usually longer than source-code line length.) A byte count that is too large will jump to the binary-insert routine at line 114. Line numbers up to 65535 will output whether they are actual line numbers or a chance byte-pair in binary. A test of number magnitude was included in an earlier version but then disregarded due to the large number of starting prompts.

Remark checking in lines 30 to 32 is part of the blank-line separation for REMs. Removing separation would delete all but the "D=0" statement; D must remain for line number printing.

Statements begin parsing in line 34. ASCII characters are restored for printing but control characters are uppercase between vertical bars. Source code rubouts are included to fill out lines in certain programs.²

Token parsing begins at line 39 with a test for unused tokens. The added space to the gather string at line 40 prevents a crash during a binary code test; a rare condition, but it was found in two listings.

Three programs were found with a FOR loop starting on an IF-true condition. Line 42 solves indenting and restoration on this rare case. Integer normally executes only one IF-true condition but, apparently, a FOR loop will execute until completed.

The printout indent restoration of statements such as "NEXT J,K" is solved by the search routine in lines 45 to 49. Of several comma tokens, only decimal value 90 is the comma in a multiple-variable NEXT statement. This search and find will restore global indenting of FOR loops. It can also be patched into the original LISZTER to solve an oversight.⁷

Some tokens allow following numeric constants. Integer BASIC flags a numeric constant with a \$B0 to \$B9 prefix (ASCII numbers 0 to 9). The test in lines 51 and 52 check for token and prefix, ignoring the prefix if it exists.

Line 53 builds the numeric constant string and gathers it in G\$. Flow must return to line 34 afterwards. The next byte can be either a token or a char-

acter; variable names are ASCII characters.

The Final Print Line

Lines 55 to 80 form the output print line, splitting and indenting as in the original LISZTER. First-priority split is still a space, but second-priority split has a vertical bar added to line 69. Control characters seem to be used more in Integer. At this point they have been converted to upper-case letters between bars and will not upset printer control.

The complex print statement group in line 77 is solely for the italics capability of the Epson printer. A single-character-set printer can substitute a simple "PRINT M\$; C\$" for both GOSUBs and PRINTs.

Possible Binary?

An IF-true test at lines 26 or 39 indicates something is wrong with the Integer source code. More than likely it is due to embedding binary code with integer. The routine at lines 114 to 120 checks this condition.

Variable LA is made up of the address of each new source line number start. That address is converted to hexadecimal and printed with the "Possible Binary From" indicator. A search now begins for any byte group meeting the following: the group is below HIMEM, the group is less than 128 bytes long, and the end-of-line byte value is found from the first-byte address plus value. A successful search will print the byte group *last* address in hex to complete the indicator, then return to line 25 for a new source line number.

The indicator may be printed several times before a correct source line is found. The number of prints will be dependent on binary content but a correct Integer source line will always follow embedded binary.

A possibility is a bit error in memory that can yield another possible binary print line. An advantage is that a printout will show beginning and ending addresses for closer examination.

An "attached" binary program will terminate at highest available memory. The possible binary last print will indicate this as \$95FF with standard DOS.

Alternatives

A purely Integer version of ILISZTER can be written by translation of the general structure. Page zero locations \$69 through \$6D can be used for

(continued)

```
105 HOME : INVERSE : PRINT " SET PAPER TO TOP OF FORM ": PRINT "
      THEN      ": PRINT "      TURN ON PRINTER      ": NORMAL : PRINT
      : GET AS
106 REM SET SCREEN WIDTH, TURN ON PROPER PORT
107 HOME : POKE 33,30: PR# 1
108 REM CONTROL CHARACTERS FOR MX-80 WITH "GRAPPLER" CARD. CHR$(9)=CTRL
      -I, CHR$(27)=ESC
109 PRINT CHR$(9)"82N" CHR$(27)"0" CHR$(9)"I"
110 REM
111 REM SET-UP TO START FIRST PRINT PAGE
112 LC = 6:PC = 1:D = 0: GOSUB 11: GOTO 25
113 REM POSSIBLE-BINARY INSERT/ADDITION ROUTINE
114 RF = 1: GOSUB 18:L = LA: GOSUB 21: PRINT M$;LB$;" >>> Possib
      le Binary from ";AS;" to ";
115 IF P > PE GOTO 121
116 IF B > 127 THEN GOSUB 2: GOTO 115: REM BYTE-COUNT TOO LARGE
117 PT = P + B - 1:BT = PEEK (PT): IF PT > PE GOTO 121
118 IF BT < > 1 OR B < 5 THEN GOSUB 2: GOTO 115: REM NO E-O-L OR BYTE-
      COUNT TOO SMALL
119 IF LA = (P - 1) THEN GOSUB 2: GOTO 115: REM AVOID REPETITION; SOMEH
      OW THE POINTER DIDN'T ADVANCE
120 P = P - 1:L = P: GOSUB 21: PRINT AS:D = 0:G$ = "": GOTO 25: REM RETUR
      N TO LINE-NUMBER START
121 L = PE: GOSUB 21: PRINT AS
122 REM ENDING ROUTINE
123 GOSUB 4: GOSUB 17: PRINT M$;LB$;"End of Listing"
124 REM OPTIONAL STATISTICS
125 GOSUB 4: PRINT M$;"Program Length = ";(PE - PS);" Bytes, Total of "
      ;TN;" Line Numbers": GOSUB 4: PRINT M$;(TS - TR);" Total Non-Rem Sta
      tements, ";TR;" Total Remarks"
126 REM TURN OFF PRINTER, RESET SCREEN AND SHOW COMPLETION
127 PR# 0: POKE 33,40: HOME : VTAB 12: HTAB 10: INVERSE : PRINT " END OF
      ILISZTING ": NORMAL : END
128 REM "ILISZTER" program to re-format INTEGER BASIC listing prints
129 REM by Leonard H. Anderson Version 2.8.8, 15 May 1982
130 REM lower case and italics for MX-80 & "GRAFTRAX"
131 REM Possible-Binary routines added to 2.8.1 (21 March 1982)
132 REM
133 REM DESCRIPTION OF VARIABLES:
134 REM
135 REM AS TEMPORARY STRING, PARTLY FOR HEX CONVERSION
136 REM B PROGRAM BYTE VALUE IN DECIMAL
137 REM BB$ 'BIG BLANK' STRING OF 48 SPACES
138 REM BC BYTE-COUNT OF A LINE, DECIMAL
139 REM BT TEMPORARY PROGRAM BYTE VALUE IN DECIMAL
140 REM CF "IF" FLAG: SET ONLY ON "IF" FOLLOWED BY "FOR"
141 REM CS CHARACTER AND TOKEN STRING TO BE PRINTED
142 REM D TEMPORARY, PARTLY FOR 'DIRECTION'
143 REM E TEMPORARY, PARTLY FOR SPLIT-LINE LIMITS
144 REM FF "FOR" FLAG: 1 = "FOR" STARTED, 0 = NO "FOR"
145 REM FS "FOR" INDENT SPACE COUNTER
146 REM GS 'GATHER' STRING TO BUILD A STATEMENT
147 REM HS HEADER ARRAY FOR PRINT-PAGE TITLE
148 REM IM INDENT SPACE MULTIPLIER
149 REM K TEMPORARY
150 REM L TEMPORARY, PARTLY FOR LOW-BYTE VALUE
151 REM LA LINE NUMBER BEGINNING ADDRESS
152 REM LC LINE COUNTER FOR PAGINATION
153 REM LL LINE-LENGTH CONSTANT
154 REM LB$ 'LITTLE BLANK' STRING OF 8 SPACES
155 REM M$ LEFT MARGIN SPACING STRING
156 REM NS LINE NUMBER STRING
157 REM P POINTER TO PROGRAM BYTE, DECIMAL
158 REM PC PAGE COUNTER FOR PRINT-PAGE HEADER
159 REM PE INTEGER PROGRAM END ADDRESS, DECIMAL
160 REM PS INTEGER PROGRAM START ADDRESS, DECIMAL
161 REM PT TEMPORARY POINTER TO PROGRAM BYTE, DECIMAL
162 REM RF "REM" FLAG: 1 = "REM" STARTED, 0 = NO "REM"
163 REM RS "REM" INDENT SPACE COUNTER
164 REM SF SPLIT-LINE FLAG: SET IF PRINT LINE MUST BE SPLIT
165 REM SS SINGLE-SPACE STRING
166 REM TN TOTAL LINE NUMBER COUNTER
167 REM TR TOTAL REMARKS COUNTER
168 REM TS TOTAL STATEMENTS COUNTER
169 REM T$ TOKEN STRING ARRAY
170 REM V ARRAY FOR TOKEN EVALUATION:
171 REM 0 = NO BINARY NUMBER FOLLOWS TOKEN
172 REM 1 = A 3-BYTE BINARY NUMBER FOLLOWS
173 REM 2 = UNUSED/INTERNAL, DO NOT PRINT
174 REM X$ HEX CHARACTER STRING FOR CONVERSIONS
```

Make ILISZT

```

200      *   TEXT FILE GENERATOR FOR "ILISZT"
210      *   VERSION 3.0, 16 APRIL 1982  LHA

220  D$ = "|D|"
230  Print D$;"OPEN ILISZT"
240  Print D$;"WRITE ILISZT"

250      *   MAKE INTEGER LOWMEM POINTER HOLD ENDING OF INTEGER PROGRAM

260  Print "POKE74,PEEK(76)"
270  Print "POKE75,PEEK(77)"

280      *   MAKE INTEGER HIMEM POINTER HOLD START OF INTEGER PROGRAM

290  Print "POKE76,PEEK(202)"
300  Print "POKE77,PEEK(203)"
310  Print "RUN ILISZTER"
320  Print D$;"CLOSE"
330  End

```

pointer re-arrangement as in the LISZT predecessor. Total code will probably exceed the 4.5K bytes of a "REM-less" ILISZTER in Applesoft. MAKE ILISZT can be either language; the created text file will be the same.

ILISZTER has successfully handled a 23K Integer program printout plus one program with two embedded binary code sections.

References

1. Apple Pugetsound Program Library

Exchange "public domain" disks (members only). Printouts of 1057 programs fill three large loose-leaf notebooks; about a quarter are Integer.

2. "Higher Text" by Ron and Darrell Aldrich, *Call —A.P.P.L.E.* version. One Integer program has two binary embeddings.
3. *MICRO on the Apple*, Volume 1, MICRO INK, pages 198-203.
4. *PEEKing at Call —A.P.P.L.E.*, Volume 2, pages 44-61, Apple Puget-

sound Program Library Exchange, 1979.

5. *What's Where in the Apple?*, William F. Luebbert, MICRO INK. For address locations only.
6. "The Inspector," Omega Micro-ware, Inc., is one example of a disk or memory byte-changer utility. Although the author has upper-/lower-case conversion on the keyboard, this utility was used to correct typos in ILISZTER's DATA statements.
7. "LISZT with Strings," Richard F. Searle, Don Cohen, Leonard H. Anderson, MICRO, May 1982, listing 2 on page 41. The easiest patch is a GOSUB in line 45 just after the "CF=1" statement; the subroutine would look for a delimiter comma in ASCII, such as "BT=44", to decrement the FOR spacer.

You may contact Mr. Anderson at 10048 Lanark St., Sun Valley, CA 91352.

MICRO

EVER WONDER HOW YOUR APPLE II WORKS?

QUICKTRACE will show you! And it can show you WHY when it doesn't!

This relocatable program traces and displays the actual machine operations, while it is running and without interfering with those operations. Look at these FEATURES:

Single-Step mode displays the last instruction, next instruction, registers, flags, stack contents, and six user-definable memory locations.

Trace mode gives a running display of the Single-Step information and can be made to stop upon encountering any of nine user-definable conditions.

Background mode permits tracing with no display until it is desired. Debugged routines run at near normal speed until one of the stopping conditions is met, which causes the program to return to Single-Step.

QUICKTRACE allows changes to the stack, registers, stopping conditions, addresses to be displayed, and output destinations for all this information. All this can be done in Single-Step mode while running.

Two optional display formats can show a sequence of operations at once. Usually, the information is given in four lines at the bottom of the screen.

QUICKTRACE is completely transparent to the program being traced. It will not interfere with the stack, program, or I/O.

QUICKTRACE is relocatable to any free part of memory. Its output can be sent to any slot or to the screen.

QUICKTRACE is completely compatible with programs using Applesoft and Integer BASICs, graphics, and DOS. (Time dependent DOS operations can be bypassed.) It will display the graphics on the screen while **QUICKTRACE** is alive.

QUICKTRACE is a beautiful way to show the incredibly complex sequence of operations that a computer goes through in executing a program.

Price: \$50

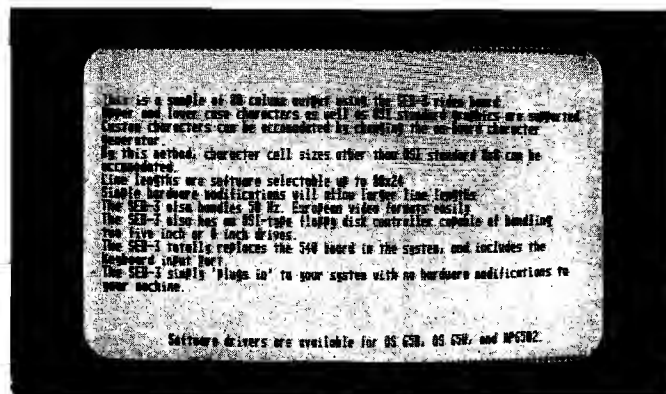
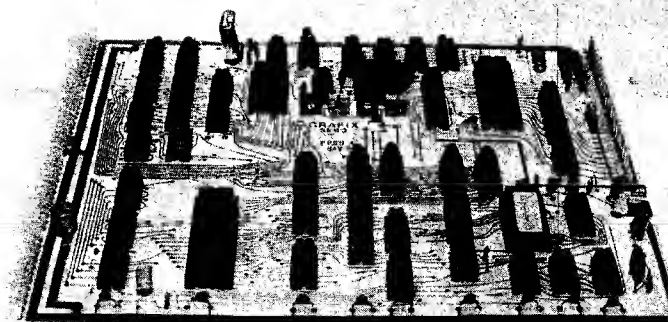
QUICKTRACE was written by John Rogers.
QUICKTRACE is a trademark of Anthro-Digital, Inc.

QUICKTRACE requires 3548 (\$E00) bytes (14 pages) of memory and some knowledge of machine language programming. It will run on any Apple II or Apple II Plus computer and can be loaded from disk or tape. It is supplied on disk with DOS 3.3.

QUICKTRACE DEBUGGER

Last address		Disassembly	
Last Instruction	FF69- A9 AA	LDA	#\$AA
Top seven bytes of stack			
Stack	ST=7C A1 32 D5 43 D4 C1 NV=BDIZC 0000=4C	Processor codes	User defined location & Contents
Accumulator X reg. Y reg. Stack pointer Processor status Content of referenced address			
Contents	A=AA X=98 Y=25 SP=F2 PS=10110001 []=DD		
Disassembly		Reference address	
Next Instruction	FF6B- B5 33	STA	\$33 [\$0033]

Anthro-Digital, Inc.
P.O. Box 1385
Pittsfield, MA 01202
413-448-8278



SUPER OSI EXPANSION BOARDS

Tired of trying to run your word processor or your DMB on an OSI 64 character video screen? Now there's the SEB-3, THE most versatile 80x24 video board anywhere is available for OSI 48 pin BUSS systems. No longer will you have to consider converting your video-based system to a serial terminal because you've found 64 characters stifling for serious business use. Nor need you give up compatibility with any existing graphics software because the SEB-3 allows you to choose ANY screen format up to 80x24 including 32x32 and 64x32. Since the SEB-3's screen format can be changed at any time under software control, even gaming displays can benefit from screens custom tailored to the game itself. The SEB-3 is so well designed and so versatile that it will not need to be replaced — ever. Simple changes in software and/or hardware will allow the SEB-3 to generate displays up to 256

columns; handle 50 Hz European formats, accommodate custom characters or character cell sizes larger or smaller than 8x8 and transparently access the screen to eliminate screen "glitches". In short, the SEB-3 will meet any demands your system may place on it now and in the future. The SEB-3 also supports an OSI-style floppy disk interface which can handle two 5" or 8" drives. Like all of the boards in the SEB series, the SEB-3 simply "plugs in" to your machine — there are absolutely NO hardware changes. The SEB-3 is designed to replace your outmoded 540 board so you don't even lose a backplane slot. Your keyboard input now also plugs into the SEB-3 — load one of the software drivers and you're ready to go!

**SEB-3 Assembled \$259.00
Kit \$220.00**

**Bare Board \$59.00
Manual only \$5.00**



If your Challenger can't generate displays like those shown above **WHAT ARE YOU WAITING FOR?** The SEB-1 High Resolution Graphics and Memory Board (for C1P and Superboard II) and the SEB-2 High Resolution Graphics and Disk Controller Board (for C2/4/8) simply 'plug-in' to your computer and give you instant access to over 49000 individually addressable pixels in up to 8 colors! Your Hi-Res screen can go from 32 x 16 alphanumerics to 256 x 192 point graphics in 11 software selectable modes. The standard video of your computer is left intact, so that none of your current software library is outmoded. Use the graphics for Business, Scientific, Education, or Gaming displays that were impossible — until now!

	SEB-1	SEB-2
Assembled and Tested	\$249.00 (5K RAM)	\$239.00 (1K RAM)
Kit	\$165.00 (No RAM)	\$199.00 (No RAM)

Installation of either board requires absolutely NO modification of your computer—they just 'plug-in'. Nor do they preclude your using any other OSI-compatible hardware or software. In addition to the Hi-Res Graphics the SEB-1 gives C1 & Superboard II users 16K of additional user memory (over and above that memory devoted to the graphics), two 16 bit timers/counters, an on-board RF modulator, and a parallel port with handshaking. The SEB-2 gives OSI 48-pin BUS users an OSI hardware/software compatible Disk controller, and an RF modulator that can be user-populated.

FOR OSI 1P, 2-4P, 2-8P, C4P, C8P

	SEB-1	SEB-2
Bare Board & Manual	\$ 59.00	\$ 59.00
Manual only	\$ 5.00	\$ 5.00

COMING: SEB-3 80 x 24 Video/Disk Controller (C2/4/8), SEB-4 48K Memory RAM/ROM (C2/4/8), SEB-5 8K RAM/Disk/Sound/Clock/Voice (C1 & Superboard).

Write for **FREE** catalog
International Requests please
supply 2 International Response Coupons

ORION



SOFTWARE ASSOC.

P.O. BOX 310, OSSINING, NY 10562



VISA

914-762-5636

BASIC Macro Function for Cursor Control

by Kerry Lourash

BASIC Macro is a machine-language program similar in function to the macro option of some assemblers. It enables Cursor Control users to insert often-used statements with only two keys when typing BASIC programs. ERGO, a routine for all C1P users, eliminates the graphic character in error messages.

BASIC Macro and ERGO

require:
OSI C1P

As a C1P owner, I type in a lot of BASIC programs, mainly because neither OSI nor independent vendors have the programs I want. While I pounded my fingers to the bone and cursed my two-fingered typing speed, I wished for a utility similar to the macro function of some assemblers. After punching out "GOSUB8000:GOTO650" for the 20th time in a program, I was inspired to write BASIC Macro.

Macro is an extension of the Cursor Control program [MICRO 36:75]. It lets you insert one of ten macros up to 70 characters long in a BASIC line with only two keystrokes (three, if you count CTRL R as two keys). If a phrase (such as GOSUB8000:GOTO650) occurs frequently in a program you're typing, store it in a BASIC line 0-9 (1 GOSUB8000:GOTO650). Now, as you encounter that phrase, hit CTRL R. A white block will appear. Type '1' and the phrase will be printed on the screen and stored in the input buffer. Should you type a line number that doesn't exist, Macro will wait for another number. If you type a letter, Macro assumes you've changed your mind about calling a macro, and exits. CTRL R stands for repeat.

When designing Macro, I had plans for a sophisticated phrase storage area with variable-length storage space. After I'd written the code to find and print the phrases, which was the lesser half of the program, I found that I'd used over half a page of memory. This approach was going to cost me well over the page of memory I had allotted for program and storage space! So I let BASIC keep track of the phrases.

To patch Macro into Cursor Control, change the input routine PATCH

at location \$1E0F to JMP \$0222 instead of JMP \$1E12.

Macro finds the BASIC line you specify, prints it on the screen, and stores it in the input buffer. If the addition of the phrase makes the line too long, the 'BEL' character is printed. To use BASIC lines 0-9 as storage space, it was necessary to teach Macro how to convert tokens to keywords, but the final program is still much shorter than my first attempt. The WINDUP routine finds the buffer count in the stack,

BASIC Macro Listing

```

10 0000      ;BASIC MACRO FOR CC
20 0000      PATCH=$1E0F
30 0000      OK=$1F10
40 0222      *=$0222
50 0222 C912  MACRO  CMP  ##12      ;CTRL R?
60 0224 D061      BNE  RESUME
70 0226 20101F    JSR  OK          ;PRINT WHITE BLOCK
80 0229 2000FD    MAC  JSR  $FD00    ;GET MACRO NUMBER
90 022C C93A      CMP  ##3A        ;IF NOT A NUMBER
100 022E B057     BCS  RESUME       ;THEN EXIT
110 0230 C930     CMP  ##30
120 0232 9053     BCC  RESUME
130 0234 E930     SBC  ##30        ;ASCII TO BINARY
140 0236 B511     STA  $11         ;LOOK FOR LINE #
150 0238 A700     LDA  #0
160 023A B512     STA  $12
170 023C 2032A4   JSR  $A432
180 023F 90E8     BCC  MAC        ;TRY AGAIN
190 0241      ;
200 0241 A003     LDY  #3          ;TO START OF LINE
210 0243 C8      FOUND  INY        ;NEXT CHAR.
220 0244 B497     STY  $97         ;SAVE Y REGISTER
230 0246 B1AA     LDA  ($AA),Y    ;GET CHAR.
240 0248 F035     BEQ  WINDUP      ;QUIT IF NULL
250 024A 3007     BMI  TOKEN      ;CONVERT IF TOKEN
260 024C A497     FND  LDY  $97    ;RESTORE Y REGISTER
270 024E 206F02   JSR  STORE
280 0251 D0F0     BNE  FOUND      ;BRANCH ALWAYS
290 0253      ;
300 0253 38      TOKEN  SEC        ;FIND & CONVERT TOKEN
310 0254 E97F     SBC  ##7F        ;TOKEN MINUS 7F
320 0256 AA      TAX          ;TOKEN INDEX IN A REG
330 0257 A0FF     LDY  ##FF
340 0259 CA      TO  DEX
350 025A F008     BEQ  T2         ;FOUND TOKEN IN TABLE?
360 025C C8      T1  INY          ;NO, NEXT LETTER
370 025D B984A0   LDA  $A0B4,Y
380 0260 10FA     BPL  T1         ;LOOP & GET NEXT CHAR.
390 0262 30F5     BMI  TO        ;LOOP TO NEXT TOKEN
400 0264 C8      T2  INY
410 0265 B984A0   LDA  $A0B4,Y    ;GET LETTER
420 0268 30E2     BMI  FND        ;LAST LETTER OF TOKEN?
430 026A 206F02   JSR  STORE
440 026D D0F5     BNE  T2
450 026F      ;
460 026F A60E     STORE  LDX  $0E  ;STORE CHAR. IN BUFFER

```

where it was stored at the start of the INPUT routine (the X register). Location \$0E, the screen character counter, is loaded into the stack to update the buffer count.

For those unfortunates who have not been converted to Cursor Control, I whipped up a short patch to the stock output routine that prints C1P error messages correctly. As the output routine prints characters on the screen, ERGO checks every carriage return to see if it comes from the error message routine. If so, ERGO steps in and prints the second letter of the error message as a letter, not a graphics character. The stock carriage return/line feed is omitted to save space on the screen. To patch ERGO into the output routine, change the contents of the output vector to the start of ERGO (\$021A=22, \$021B=02).

You may contact Kerry Lourash at 1220 North Dennis, Decatur, IL 62522.

MICRO

BASIC Macro Listing (Continued)

470 0271 E047	CFX	#\$47	
480 0273 B005	BCS	ST0+1	
490 0275 297F	AND	#\$7F	;\$ZERO HI BIT
500 0277 9513	STA	\$13,X	
510 0279 2CA907	STO	BIT	\$07A9 ;\$BEL CHAR. IF >71
520 027C 4CE5A8	JMP	\$ABE5	;\$PRINT CHAR.
530 027F	;		
540 027F BA	WINDUP	TSX	;\$UPDATE BUFFER COUNT
550 0280 A50E	LDA	\$0E	;\$LINE COUNT IN STACK
560 0282 9D0201	STA	\$0102,X	
570 0285 A901	LDA	#1	;\$NON-PRINTING CHAR.
580 0287 4C121E	RESUME	JMP	PATCH+3 ;\$BACK TO CC

ERGO Listing

10 0000	;	ERGO ROUTINE	
20			
30 0222	*\$0222		
40 0222 C90D		CMP	#13 IS CHAR A OR ?
50 0224 D015		BNE EXIT	
60 0226 8650		STX	\$50 SAVE X REG.
70 0228 BA		TSX	GET STACK POINTER
80 0229 BD0501		LDA	\$105,X CALLING ADDRESS \$A252?
90 022C C952		CMP	#\$52
100 022E D007		BNE	NOERR
110 0230 BD0601		LDA	\$106,X
120	0233 C9A2	CMP	#\$A2
130 0235 F007		BEQ	ERGO YES, PRINT ERR MESS.
140 0237 A650	NOERR	LDX	\$50 RESTORE A&X REGS.
150 0239 A90D		LDA	#13
160 023B 4C69FF	EXIT	JMP	\$\$\$F69 TO REGULAR OUTPUT
170			
180 023E A650	ERGO	LDX	\$50 RESTORE X REG.
190 0240 20E3A8		JSR	\$A8E3 PRINT '?'
200 0243 BD64A1		LDA	\$A164,X FIND 1ST LETTER
210 0246 20E5A8		JSR	\$A8E5 PRINT IT
220 003F BD65A1		LDA	\$A165,X FIND 2ND LETTER
230 024C 297F		AND	#\$7F ZERO HI BIT
240 024E 4C5FA2		JMP	\$A25F TO REG. ERR ROUTINE

OSI

Stankiewicz & Robinson,
authors of MINOS, NIGHT RIDER, etc.,
proudly present to you:

C1P

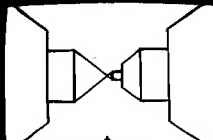
34 original **PROGRAMS** on tape all for the unbelievably low price of **\$29.95!!**
That's less than \$1 each!



PINBALL

ARCADE TYPE

NIGHT RIDER*
COSMIC DEBRIS*
MINOS*
STREET SWEEPERS
RIDGE CRUISER
CAGE*
PINBALL
OSI GRAND*
MINE FIELD
WORM
DEPTH CHARGE
GOTCHA!



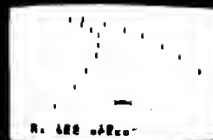
MINOS (MAZE)

STRATEGY

TAKE FOUR
MIMIC
MANGALA
NEIGHBORS
BAR
LIFE FOR TWO*

KALEIDOSCOPIC

LIVING PATTERNS
KALLIDSCOPE
DRAW ME



NIGHT RIDER

UTILITIES

TAPE VERIFIER
LISTING LINE RE*
VERSATILE LINE RE*
FINE LOCATOR

STATISTICS

CHI SQUARE
FUNCTION PLOTTER
BETTER RND* GEN
PROBABILITY*1



RIDGE CRUISER

MISCELLANEOUS

MESSAGE ENCODER
TYPING TUTOR
PHONE NUMBER
DEHYDRATION
BLACK JACK DRILL

(*Previously sold
by AARDVARK V)



All programs will run on 8k C1P.
Many are compatible for C2-4
and run in 4k

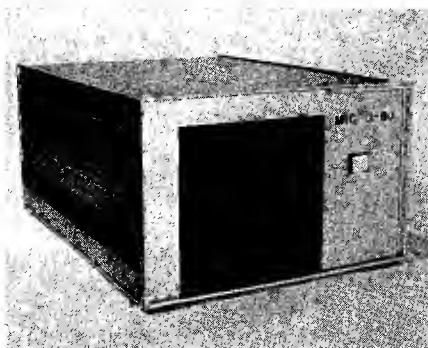
VICTORY

SOFTWARE

Please add \$1.50 postage & handling
PA resident please add 6% sales tax
Charge customers include * and
expiration date

VICTORY SOFTWARE CORP.
2027-A S.J. RUSSELL CIRCLE
ELKINS PARK, PA 19117
(215) 576-5625

NEW FROM D & N MICRO PRODUCTS, INC.



MICRO-80 COMPUTER

Z80A CPU with 4MHz clock and CP/M 2.2 operating system. 64K of low power static RAM. Calendar real time clock. Centronics type parallel printer interface. Serial interface for terminal communications, dip switch baud rates of 150 to 9600. 4" cooling fan with air intake on back of computer and discharge through ventilation in the bottom. No holes on computer top or side for entry of foreign object. Two 8" single or double sided floppy disk drives. IBM single density 3740 format for 243K of storage on each drive. Using double density with 1K sectors 608K of storage is available on a single sided drive or 1.2 meg on a double sided drive. Satin finish extruded

aluminum with vinyl woodgrain decorative finish. 8 slot backplane for expansion. 48 pin buss is compatible with most OSI boards. Uses all standard IBM format CP/M software.

Model 80-1200 \$2995
2 8" single sided drives, 1.2 meg of storage
Model 80-2400 \$3495
2 8" double sided drives, 2.4 meg of storage
Option 001 \$ 95
Serial printer port, dip switch baud rate settings

Software available in IBM single density 8" format.

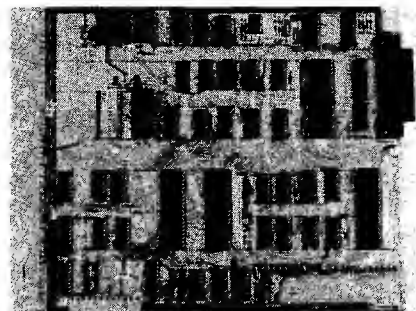
Microsoft		Digital Research		Micropro	
Basic-80	\$289	PL/1-80	\$459	Wordstar	\$299
Basic Compiler	\$329	Mac	\$ 85	Mail-Merge	\$109
Fortran-80	\$410	Sid	\$ 78	Spellstar	\$175
Cobol-80	\$574	Z-Sid	\$ 95	SuperSort I	\$195
Macro-80	\$175	C Basic-2	\$110	Pascal	
Edit-80	\$105	Tex	\$ 90	Pascal/MT +	\$429
Mu Simp/Mu Math	\$224	DeSpool	\$ 50	Pascal Z	\$349
Mu Lisp-80	\$174	Ashton-Tate		Pascal M	\$355
		dBase II	\$595		

Convert almost any static memory OSI machine to CP/M® with the D & N-80 CPU Board.

Z80A CPU with 4MHz clock. 2716 EPROM with monitor and bootstrap loader. RS-232 serial interface for terminal communications or use as a serial printer interface in a VIDEO system. Disk controller is an Intel 8272 chip to provide single or double density disk format. 243K single density or 608K double density of disk storage on a single sided 8" drive. A double sided drive provides 1.2 meg of storage. DMA used with disk controller to unload CPU during block transfers from the disk drives. Optional Centronics type parallel printer port com-

plete with 10 ft. cable. Optional Real Time Calendar Clock may be set or read using 'CALL' function in high level languages. Power requirements are only 5 volts at 1.4 amps. Available with WORDSTAR for serial terminal systems. INCLUDES CPM 2.2

D & N-80 serial	\$695
D & N-80 serial w/Wordstar	\$870
D & N-80 video	\$695
Option 001	\$ 80
parallel printer and real time calendar clock	



D & N-80 CPU BOARD

OTHER OSI COMPATIBLE HARDWARE

IO-CA10X Serial Printer Port \$125
Compatible with OS-65U and OS-65D software
IO-CA9 Parallel Printer Port \$175
Centronics standard parallel printer interface with 10 ft. flat cable
BP-580 8 Slot Backplane \$ 47
Assembled 8 slot backplane for OSI 48 pin buss
24MEM-CM9 \$380 **24MEM-CM9F** \$530
16MEM-CM9 \$300 **16MEM-CM9F** \$450
8MEM-CM9 \$210 **8MEM-CM9F** \$360
BMEM-CM9F \$ 50 **FL470** \$180
24K memory/floppy controller card supports up to 24K of 2114 memory chips and an OSI type floppy disk controller. Available fully assembled and tested with 8, 16, or 24K of memory, with floppy controller (F). Controller supports 2 drives. Needs separated clock and data inputs. Available Bare (BMEM-CM9F) or controller only (FL-470). Ideal way to upgrade cassette based system

C1P-EXP Expansion Interface \$ 65
Expansion for C1P 600 or 610 board to the OSI 48 pin buss. Requires one slot in backplane. Use with BP-580 backplane
BIO-1600 Bare IO card \$ 50
Supports 8K of memory, 2 16 bit parallel ports may be used as printer interfaces. 5 RS-232 serial ports, with manual and Molex connectors
DSK-SW Disk Switch \$ 29
Extends life of drive and media. Shuts off minifloppy spindle motor when system is not accessing the drive. Complete KIT and manual

Disk Drives and Cables
8" Shugart SA801 single sided \$395
8" Shugart SA851 double sided \$585
FLC-66ft. cable from D & N or OSI controller to 8" disk drive \$ 69
5 1/4" MPI B51 with cable, power supply and cabinet \$450
FLC-5 1/48 ft. cable for connection to 5 1/4 drive and D & N or OSI controller, with data separator and disk switch \$ 75
Okidata Microline Printers
ML 82A Dot Matrix Printer \$534
120 CPS, 80/120 columns, 9.5" paper width, friction or pin feed
ML 83A Same as 82A except \$895
16" paper width, 132/232 columns with tractor feed
ML 84 Same as 82A except 200 CPS, \$1152
16" paper width, 132/232 columns, 2K buffer, dot addressable graphics, with tractor feed

D & N Micro Products, Inc.
3684 N. Wells St.
Fort Wayne, Ind. 46808
(219) 485-6414



TERMS \$2.50 shipping, Foreign orders add 15%.
Indiana residents add 4% sales tax.

ATARI Character Graphics from BASIC, Part 3

by Paul Swanson

You can remove the screen flicker by adding a short machine-language program to Atari's vertical blank interrupt routine.

Character Graphics
requires:
Atari 400/800

Last month I explained how to enable and use Atari's fine scrolling function (:). The only big problem was that the screen flickered a little because you had to shut off ANTIC, along with the display, in order to alter the horizontal scroll register.

There are several registers like that—you can't write to them while ANTIC is displaying a screen or you get strange effects. Most of these are taken care of by shadowing. However, the horizontal scroll register is not shadowed, so we need a different technique.

Shadowing

Shadowing is a method of updating video-related registers without interrupting the display in progress. Certain memory locations ("shadow" registers) are set aside to represent the actual video registers. When ANTIC completes the job of displaying one screen, it sends an interrupt signal to the 6502. Since ANTIC is not doing anything but waiting for the electron beam to return to the upper left corner of the screen, the 6502 has time to execute many instructions. Among the things accomplished during this vertical blank period is an update of the actual video registers from the contents of the shadow registers. This guarantees that all of the hardware registers are written while ANTIC is not drawing on the screen. At the end of the interrupt routine, the 6502 automatically returns to whatever it was doing before the interrupt occurred, so this process is almost invisible to the main program. This in-

terrupt routine happens at the end of every sweep of the electron beam, or exactly sixty times per second.

The Vertical Blank Interrupt Routine

Every sixtieth of a second your program, whether in BASIC or machine language, gets interrupted for this special routine. Actually, there are two routines. The first one, which almost always runs, is called the immediate vertical blank interrupt routine. It takes care of all of the timers in the system, which includes the real time clock in locations 18 through 20

[decimal]]. It adds one each frame so that $\text{PEEK}(20) + \text{PEEK}(19) * 256 + \text{PEEK}(18) * 65536$ always reveals the elapsed time in sixtieths of a second.

The second routine is tacked on to the end of the first one. This second part is called the deferred vertical blank interrupt routine. You can easily stop this routine from running by setting the critical flag [a 1 into location 66]. In addition to writing the shadowed information to the hardware registers, this second part also updates a few other timers, maintains the keyboard auto-repeat and debounce functions, and reads and interprets the game controllers into special memory locations.

By altering two vector locations, you can replace or add to the existing interrupt routines. Each vector is a two-byte address stored in low, high order.

The vertical blank interrupt starts with a signal generated by ANTIC at the end of the display. This signal can be masked by the hardware register NMIEN [decimal location 54286]. If the contents last written here were 64,

Listing 1: Routine to shadow the fine scrolling registers. The JMP location xxxx will be the vector value at location \$224. The shadow registers will be at locations \$610 and \$611.

```
0600 AD 11 06 LDA $611
0603 8D 05 D4 STA $D405
0606 AD 10 06 LDA $610
0609 8D 04 D4 STA $D404
060C 4C xx xx JMP $xxxx
```

Listing 2

```
1 REM *** Custom Character Set ***
2 REM *** Vertical Blank ***
3 REM *** Interrupt routine ***
4 REM
5 REM *** Program by... ***
6 REM *** Paul S. Swanson ***
7 REM
8 REM
9 REM --- Calc. position in mem. ---
10 DIM S$(1024)
20 A=ADR(S$)
30 B=INT(A/512+1)*2
40 CBASE=B*256-A+1
47 REM
48 REM
49 REM --- Clear S string ---
50 S$(1)=CHR$(0)
60 S$(1024)=CHR$(0)
70 S$(2)=S$(1)
77 REM
78 REM
79 REM --- Move standard set down ---
80 FOR I=0 TO 511
90 S$(CBASE+I,CBASE+I)=CHR$(PEEK(I+57344))
100 NEXT I
107 REM
108 REM
```

(continued)

Listing 2 (continued)

```

109 REM --- Set # to character ---
110 FOR I=24 TO 31
120 READ N
130 S$(I+CBASE,I+CBASE)=CHR$(N)
140 NEXT I
147 REM
148 REM
149 REM --- GR.2 - No text window ---
150 GRAPHICS 18
152 GOSUB 500
157 REM
158 REM
159 REM --- Find Display List ---
160 DLIST=PEEK(560)+PEEK(561)*256
162 SLOC=PEEK(DLIST+4)+PEEK(DLIST+5)*256
167 REM
168 REM
169 REM --- Set scroll enables ---
170 POKE DLIST+3,PEEK(DLIST+3)+48
180 FOR I=6 TO 16
190 POKE DLIST+I,PEEK(DLIST+I)+48
200 NEXT I
207 REM
208 REM
209 REM --- Initialize position ---
210 VPOS=96
220 HPOS=80
222 POKE 756,B
224 WING=1
226 S=14
227 REM
228 REM
229 REM --- Draw character in position ---
230 V=INT(VPOS/16)
232 IF WING=1 THEN SOUND 0,10,0,6
240 VSCROL=VPOS-V*16
250 H=INT(HPOS/8)
260 HSCROL=HPOS-H*8
262 IF WING=1 THEN WING=2:S$(CBASE+25,CBASE+25)=CHR$(0):S$(
(CBASE+26,CBASE+26)=CHR$(231):GOTO 266
264 WING=1:S$(CBASE+25,CBASE+25)=CHR$(195):S$(CBASE+26,CBASE+26)
=CHR$(36)
266 P1=V*24+H
270 IF P<>P1 THEN POKE SLOC+P,0
280 POKE 1552,HSCROL
290 POKE 1553,15-VSCROL
291 IF P<>P1 THEN P=P1:FOR I=1 TO 3:NEXT I
292 POKE SLOC+P,3
294 SOUND 0,10,0,2
297 REM
298 REM
299 REM --- Read Joystick ---
300 OLDS=S:S=STICK(0)
310 IF S=15 THEN S=OLDS
320 VMOVE=0
330 HMOVE=0
340 IF S=9 OR S=13 OR S=5 THEN VMOVE=2
350 IF S=10 OR S=14 OR S=6 THEN VMOVE=-2
360 IF S>4 AND S<8 THEN HMOVE=1
370 IF S>8 AND S<12 THEN HMOVE=-1
380 IF VMOVE+VPOS>=0 AND VMOVE+VPOS<191 THEN VPOS=VPOS+VMOVE
390 IF HMOVE+HPOS>=0 AND HMOVE+HPOS<192 THEN HPOS=HPOS+HMOVE
400 IF VMOVE=2 THEN WING=2
410 GOTO 230
497 REM
498 REM
499 REM --- SET UP VBLANK ROUTINE ---
500 FOR I=1 TO 13
510 READ N
520 POKE 1535+I,N
530 NEXT I
540 POKE 66,1
550 POKE 1549,PEEK(548)
560 POKE 1550,PEEK(549)
570 POKE 548,0
580 POKE 549,6
590 POKE 66,0
600 RETURN
1000 DATA 0,195,36,24,24,36,0,0
1010 DATA 173,17,6,141,5,212,173,16,6,141,4,212,76

```

the interrupt will happen. Writing a zero will prevent the interrupt.

If the signal is not masked by NMIEN, the 6502 is interrupted and a branch to the immediate vertical blank interrupt routine occurs. This updates the real time clock, processes the attract mode, and maintains a special system timer, CDTMV1 (refer to Atari manuals).

When the immediate mode vertical blank routine is completed, the flag CRITIC (memory location 66) is checked, as is the processor interrupt bit I. If either is non-zero, the interrupt sequence is terminated with a return to the main program 6502 instruction RTI. Otherwise, the interrupt routine continues with the deferred portion.

This second part moves all the shadow registers into the hardware registers, updates a few other system timers, and decodes the results read from the game controllers. When it has finished, it branches through the vector at location 548 (decimal — 2 bytes). Unless you alter it, this location points to an RTI routine.

Every time there is a vertical blank interrupt, the computer uses the address at location 546 to find the immediate vertical blank interrupt routine. It uses the address at location 548 only when the critical flag and the I bit are not set. BASIC cannot access the I bit directly, but it can write to the critical flag with a POKE.

Your Own Routine

To shadow your fine scrolling values so that you don't interrupt the screen while it is being drawn, you must add on your own machine-language routine. This can be done by altering the pair of memory locations called VVBLKD (Vector for Vertical BLank Deferred routine — this is the one at location 548).

First you must write your routine in machine language and store it in a fixed place in memory. In the sample program, the routine requires 15 bytes and starts at location \$600 (1536 in decimal). A BASIC POKE routine may be used to install this code.

Since BASIC is so slow, you must make allowances for certain odd occurrences. What happens if a vertical blank routine tries to use a vector between the time you write one byte and the time you write the next byte? Your program crashes! To get around this potential catastrophe, you can shut the

Listing 1 shows the routine used to form shadow registers for the fine scrolling hardware registers. You must POKE the first 13 bytes into memory, then copy locations 548 and 549 into bytes 14 and 15. This causes the routine to jump to the location that the vertical blank interrupt routine normally jumps to on completion. To get

The machine-language program takes the values in locations \$610 and \$611 (decimal 1552 and 1553) and stores them into the horizontal and vertical scroll hardware registers. Then it jumps back into the vertical blank interrupt routine where we first interrupted it. Locations 1552 and 1553 (decimal) now act as shadow registers for horizontal and vertical scroll values, respectively.

Listing 2 enhances the program presented in last month's article by adding the shadowing routine. The machine-language routine is converted to decimal and included as line 1010 in a DATA statement. A new subroutine, called at line 152, has been added at line 500. It first READs the machine-language routine into the locations

There are a few other changes made to the portion that scrolls the bird. Lines 266 through 292 are altered. Line 266 now calculates the new position. If it is the same as the old position except for the scrolling values, the character is not erased. It is erased only when the position value has changed; this limits the flickering substantially.

"INTERESTING SOFTWARE"

8781 Troy St. - Spring Valley, CA 92077
(619) 466-2200

WORLD ALPHABETS

Дети едят бросимая пшеница.

Ten type fonts allow user to create text or use pronunciation tables in Arabic, Cherokee Indian, Hieroglyphics, Greek, Hebrew, Japanese, Russian, Sanskrit or Roman. Author: W.C. Jones

Diskette \$89.95

BASIC LEARNING PACKAGE

An introduction to the Apple II or II Plus Computer. Teaches beginner to program in BASIC. Author: J.J. Sudikatus

Diskette \$49.95

Both require an Apple II with Applesoft, 48K, plus disk drive. Epsom printer with Grafrax is optional.

Apple II or II Plus and Applesoft are trademarks of Apple Computer, Inc.

Lines 550 and 560 are altered to POKE into the new shadow registers. ANTIC is not turned off at all. Line 291 is added to update the position value P and cause a slight delay if the position value were changed. This delay guarantees that there has been at least one vertical blank interrupt routine since the new values were written to the shadow registers. The hardware registers are updated before line 292 is executed. Line 292 puts the bird on the screen in the position indicated by P. If the position were not altered, this line doesn't actually do anything. If the position value has been changed, it draws the bird in the new position.

There is still a slight flicker every once in awhile, but this will not be noticeable if other things are happening at the same time. The only way to eliminate the flicker altogether is to use machine language to update the bird as well. By using shadow registers you could write a vertical blank interrupt routine that would take your position values and reduce them to the

screen position and the fine scrolling values. BASIC is a much easier language in which to create programs, but a little machine language now and then can help smooth out the rough edges. If you can get away with routines as short as the one in listing 1, it is certainly worth it.

What To Do With This Information

The character graphics example here was intended for instruction only. However, the shadowing described in this article, combined with the custom character set and fine scrolling described in parts 1 and 2, needs only to be combined with a little imagination to produce some elegant software.

Paul Swanson is our Atari columnist. You may contact him at 97 Jackson Street, Cambridge, MA 02140.

MICRO™

MICRO is publishing an OSI book!

OSI users will be getting a book of their own. Early in 1983, **MICRO** magazine plans to publish a strictly **OSI** volume!

We will include a variety of topics — *BASIC Enhancements, Machine-Language Aids, Hardware, I/O Enhancements*, and a "What's Where in the OSI" reference guide. We'll supply more details soon.

Let us know what *you* would like to see in this book. Or, if you've written an article/program that you think should be a part of this volume, send it in now!

UPGRADE YOUR AIM-65* INSTANTLY

*A trademark of Rockwell Inc.

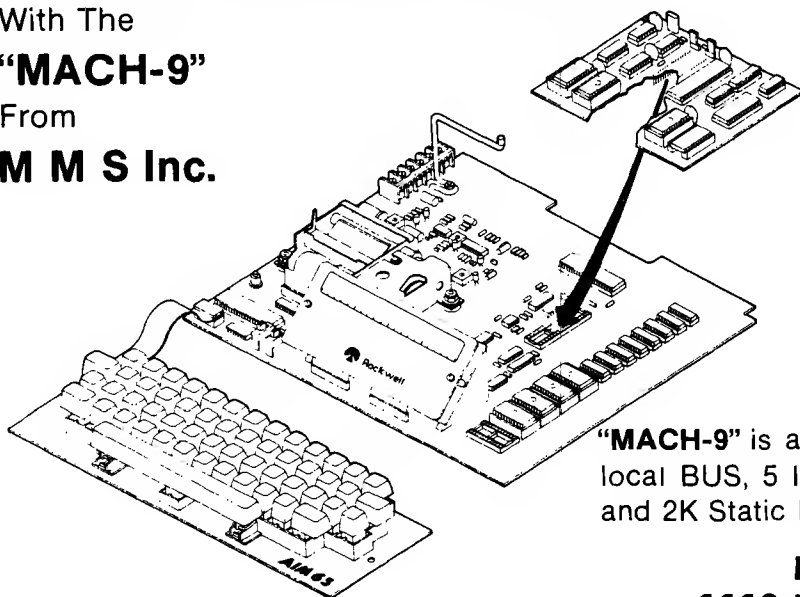
To A 6809 Development System

With The

"MACH-9"

From

M M S Inc.



INTRODUCTORY PRICE

\$239.

Plus \$6 U.P.S.
And Handling

Includes:

- *6809 CPU Plug-in Assembly
- *Super-set of AIM Monitor
- *Two-Pass Symbolic Assembler
- *Complete Monitor Source Listings
- *Enhanced Cut & Paste Editor
- *200 Page Manual
- *Full I/O Control

"MACH-9" is assembled and tested with local BUS, 5 locking low force ROM sockets and 2K Static RAM

M M S Inc.

1110 E. PENNSYLVANIA ST.

TUCSON, AZ 85714

(602) 746-0418



APPLESOFT GOTO/GOSUB Checking Routine

by Peter J.G. Meyer

This 194-byte machine-language routine will check all GOTO and GOSUB references in an Applesoft program and display any that refer to non-existent lines. The source program also demonstrates how to make use of the machine-language subroutines available in the Applesoft Interpreter.

GOTO/GOSUB Checker requires:

Apple II with Applesoft

In a previous article [MICRO 43:101] I presented a short assembly-language program for a utility that would display the bytes constituting a specified line in an Applesoft program. That utility was constructed using eight machine-language subroutines available in the Applesoft Interpreter and the Apple Monitor.

In this article I will use two of those routines [LINGET and FNDLIN] together with six others to construct a utility for checking the GOTO and GOSUB references in an Applesoft program. This utility does the useful task of going through an Applesoft program looking for GOTOs and GOSUBs. When it finds one, it searches the program for the referenced line. If the line does not exist, it displays the offending statement with the line number in which it occurs.

To understand the assembly-language program presented here, it is necessary only to understand the structure of an Applesoft line in RAM and the function of the eight Applesoft subroutines that are employed. Of course, it also helps to know a little about 6502 assembly-language programming, but novices should not be deterred.

An Applesoft program line, as it

exists as bytes in RAM, consists of four consecutive parts:

1. Two bytes containing the address of the following line (low byte then high byte, as usual).
2. Two bytes containing the line number in hexadecimal.
3. The tokenized text of the line (in which, for example, GOTO is represented by the token byte \$AB).
4. The end-of-line token, \$00.

The text of the line may consist of several statements. In this case each statement (except the last) is followed by the end-of-statement token, \$3A (which is the byte used as the ASCII representation of the colon, ':'). The

final statement in the line is followed, not by an end-of-statement token, but by the end-of-line token.

For example, suppose the program line "10 IF A = 0 THEN GOSUB 120: ON B GOTO 340,560" is the first in a program. It will (normally) occur at \$0801 and be represented in RAM as shown in figure 1.

Good programming style is simply knowing what you want to do, and stating clearly how to do it. In this case, what we want to do is as follows. For each line in the Applesoft program:

1. Inspect the line for GOTOs (\$AB tokens), THENs (\$C4 tokens), and GOSUBs (\$B0 tokens).

Figure 1

801 - 1A 08	pointer to next line
803 - 0A 00	"10" in hexadecimal
805 - AD 41 D0 30	"IF A = 1"
809 - C4 B0 31 32 30 3A	"THEN GOSUB 120:"
80F - B4 42 AB 33 34 30 2C 35 36 30	"ON B GOTO 340,560"
819 - 00	end-of-line token

Listing 1

```

2 ;*****
3 ;
4 ;*      GOTO/GOSUB CHECKER
5 ;*
6 ;*      BY PETER MEYER
7 ;*
8 ;*      APRIL 1982
9 ;*
10 ;*****
11 ;
12 ;*      APPLESOFT SUBROUTINES
13 ;
14 CHRGOT  EPZ $B1
15 CHRGOT  EPZ $B7
16 FNDLIN  EQU $D61A
17 STXTPT  EQU $D697
18 LINGET  EQU $DA0C
19 CRDO    EQU $DAFB
20 STROUT  EQU $DB3A
21 LINPRT  EQU $ED24
22 ;*      STANDARD ZERO PAGE LOCATIONS
23 ;
24 LINNUM  EPZ $50
25 TXITAB  EPZ $67
26 TXIPTR  EPZ $B8
27 ;
28 ;*      SPECIAL ZERO PAGE LOCATIONS
29 ;
30 TOKEN   EPZ $F9

```

2. If none are found, continue with the next line, until the end of the program is reached.
3. If a GOTO, THEN, or GOSUB token is found, read the line number following the token.
4. Search through the program for a line so numbered.
5. If the line is found, continue inspecting the current line for GOTOs, THENs, and GOSUBs.
6. If no such line is found, report this fact by displaying the current line number and the offending GOTO, THEN, or GOSUB statement (then continue the inspection).

To go through RAM one byte at a time, Applesoft has the subroutine CHRGET, which is located on page zero [at \$B1]. This routine makes use of the two-byte pointer called TXTPTR [at \$B8,B9]. TXTPTR is usually pointing to a byte somewhere in the Applesoft program in RAM. The effect of CHRGET is to advance TXTPTR to the next byte and to load that byte into the accumulator [setting certain flags along the way]. Thus, by repeatedly invoking CHRGET we can go through each program line looking for GOTO and GOSUB tokens. (CHRGOT, at \$B7, is CHRGET without the initial advance of TXTPTR. It simply loads the accumulator with whatever byte TXTPTR is pointing to.)

Having found a GOTO, THEN, or a GOSUB token, we can then use the subroutine LINGET [at \$DA0C] to read the line number and place it [in hexadecimal form] in the zero-page location LINNUM (\$50,51). We can use LINGET for this purpose because this is precisely what LINGET was designed to do.

To help you search through a program to find a line whose number is at LINNUM, there is the routine FNDLIN [at \$D61A]. When this routine returns, the carry flag is set if such a line was found, otherwise the carry flag is clear. In the latter case we proceed using CHRGET to look for further GOTOs and GOSUBs.

If FNDLIN returns with the carry flag set, then we have found a reference to a non-existent line and a report to this effect is in order. This report only needs to consist of 1. the number of the line containing the offending statement, 2. the word GOTO, THEN, or GOSUB, followed by 3. the number of the non-existent line referred to.

For printing numbers we have the

Listing 1 (continued)

```

31 LN1      EPZ $FA
32 LN2      EPZ $FC
33 ;
34 ;*      OTHER LOCATIONS
35 ;
36 DOS'WS   EQU $3D0          ;DOS WARM START VECTOR
37 SPEAKER  EQU $C030
38 ;
39 ;*****
40 ;
41          ORG $300          ;OR ANYWHERE CONVENIENT
42 BEGIN:
43          JSR CRDO          ;PRINT <CR>
44          ;SET TXTPTR TO BYTE PRECEDING LINK FIELD OF FIRST LINE
45          JSR STXTPT
46 NEXTLINE:
47          JSR CHRGET
48          LDY #1            ;END-OF-PROGRAM DOUBLE 00
49          LDA (TXTPTR),Y    ;REACHED YET?
50          BNE SAVLINNO      ;IF NOT
51          JSR CRDO          ;PRINT FINAL <CR>
52          JMP DOS'WS        ;BACK TO BASIC
53 SAVLINNO:
54          ;IN CASE WE NEED TO PRINT IT LATER
55          INY
56          LDA (TXTPTR),Y
57          STA LN1
58          INY
59          LDA (TXTPTR),Y
60          STA LN1+1
61          ;ADVANCE TXTPTR TO FIRST BYTE IN TEXT OF LINE
62          LDA TXTPTR
63          CLC
64          ADC #3
65          STA TXTPTR
66          BCC GOTHRULN
67          INC TXTPTR+1
68 GOTHRULN:
69          ;INSPECTING EACH BYTE IN TURN
70          JSR CHRGET
71          CMP #0            ;END-OF-LINE TOKEN?
72          BEQ NEXTLINE      ;IF SO
73          CMP #$C4          ;'THEN' TOKEN
74          BNE NEXT
75          LDY #1
76          LDA (TXTPTR),Y
77          SEC
78          SEC #$30
79          CMP #$0A
80          BCS GOTHRULN
81          LDA #$C4          ;'THEN' TOKEN
82          BNE STORE
83          CMP #$AB          ;'GOTO' TOKEN
84          BEQ STORE
85          CMP #$B0          ;'GOSUB' TOKEN
86          BNE GOTHRULN
87          STORE STA TOKEN
88 READLINNO:
89          JSR CHRGET        ;ADVANCE TXTPTR TO LINE NO.
90          JSR LINGET        ;READ LINE NO.,STORE IN LINNUM
91          LDA LINNUM
92          LDY LINNUM+1
93          STA LN2
94          STY LN2+1
95          LDA SPEAKER       ;EACH CLICK MEANS A PROG SEARCH
96          JSR FNDLIN        ;SEARCH PROGRAM FOR A LINE
97          BCS CHKCOMMA      ;IF LINE FOUND
98          LINNOTED:
99          JSR CRDO          ;PRINT <CR>
100         LDA LN1+1
101         LDX LN1
102         JSR LINTPT
103         LDA TOKEN
104         CMP #$C4          ;'THEN' TOKEN
105         BNE NEXT1
106         LDA #THEN
107         LDY /THEN
108         JMP PRINT
109         NEXT1 CMP #$B0    ;'GOSUB'
110         BEQ NEXT2
111         LDA #GOTO
112         LDY /GOTO
113         JMP PRINT
114         NEXT2 LDA #GOSUB
115         LDY /GOSUB
116         PRINT JSR STROUT ;PRINT GOTO OR GOSUB

```

Applesoft routine LINPRT (at \$ED24), which prints, in decimal form, the hexadecimal number whose high byte is in the accumulator and whose low byte is in the X-register. For printing text we have the routine STROUT (at \$DB3A), which will print the string pointed to by the Y-register (high byte) and the accumulator (low byte). (The string must be terminated by a \$00 or a \$22.)

Thus, Applesoft provides us with all the routines we need for the job. With a good assembler and some attention to detail, these can be put together to produce a machine-language routine to perform the required task. The source program in listing 1 demonstrates how this can be done.

Once assembled and BSAVED, this utility is used as follows: LOAD your program into RAM and BRUN the routine or, if it is already installed, simply CALL it. Line references in ONERR GOTOS and GOSUBs will also be checked, as will all line references (not just the first) in ON X GOTOS and GOSUBs.

Listing 1 (continued)

```

038D A5 FD      117      LDA LN2+1
038F A6 FC      118      LDX LN2
0391 20 24 ED    119      JSR LINPRT          ;PRINT LINE REFERRED TO
0394            120      CHKCOMMA:
0394            121      ;IN CASE OF MULTIPLE GOTO,OR GOSUB
0394 20 B7 00    122      JSR CHRGOT
0397 C9 2C      123      CMP #$2C          ;COMMA?
0399 F0 B3      124      BEQ READLNNO      ;IF SO
039B A5 B9      125      LDA TXTPTR+1      ;DECREMENT TXTPTR IN PREP
039D D0 02      126      BNE NEXT3        ;FOR NEXT USE OF CHRGOT
039F C6 B9      127      DEC TXTPTR+1
03A1 C6 B8      128      NEXT3      DEC TXTPTR
03A3 4C 2A 03    129      JMP GOTHRLN
03A6            130      ;
03A6            133      ;*****
03A6            134      ;*  STRINGS
03A6 20 20 20    135      GOTO      .DA '      GOTO ''
03A9 47 4F 54
03AC 4F 20 22
03AF 20 20 20    136      GOSUB      .DA '      GOSUB ''
03B2 47 4F 53
03B5 55 42 20
03B8 22
03B9 20 20 20    137      THEN      .DA '      THEN ''
03BC 54 48 45
03BF 4E 20 22
03C2            138      END

```

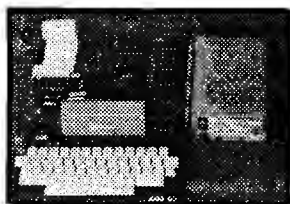
Peter Meyer is the author of Agenda Files, from Special Delivery Software, and Routine Machine, recently released by Southwestern Data Systems. He is currently designing applications software

in Europe. You may contact him at 55 Sutter St., Suite 608, San Francisco, CA 94104.

MICRO

AIM + POWER from COMPUTECH

All prices
postpaid
(Continental
U.S.-
otherwise
\$2 credit)



Check the
outstanding
documenta-
tion supplied
with AIM65!

- Top quality power supply designed to Rockwell's specs for fully populated AIM65 — includes overvoltage protection, transient suppression, metal case and power cable.
- PSSBC-A (5V 2A Reg; 24V .5A Avg, 2.5A Peak, Unreg) \$64.95
- Same but an extra AMP at 5 volts to drive your extra boards:
- PSSBC-3 (5V 3A Reg; 24V .5A Avg, 2.5A Peak, Unreg) \$74.95
- The professional's choice in microcomputers:
- AIM65/1K RAM \$429.95 BASIC (2 ROMS) \$59.95
- AIM65/4K RAM \$464.95 ASSEMBLER (1 ROM) \$32.95
- FORTH (2 ROMS) \$59.95.

SAVE EVEN MORE ON COMBINATIONS

- AIM65/1K+PSSBC-A ... \$479.95 AIM65/4K+PSSBC-3 ... \$524.95
- We gladly quote on all AIM65/40 and RM65 items as well.

ORDERS: (714) 369-1084

P.O. Box 20054 • Riverside, CA 92516
California residents add 6% sales tax



FORTH-79

Ver. 2 For your APPLE II/II+

The complete professional software system, that meets ALL provisions of the FORTH-79 Standard (adopted Oct. 1980). Compare the many advanced features of FORTH-79 with the FORTH you are now using, or plan to buy!

FEATURES	OURS	OTHERS
79-Standard system gives source portability.	YES	_____
Professionally written tutorial & user manual	200 PG.	_____
Screen editor with user-definable controls.	YES	_____
Macro-assembler with local labels.	YES	_____
Virtual memory.	YES	_____
Both 13 & 16-sector format.	YES	_____
Multiple disk drives.	YES	_____
Double-number Standard & String extensions.	YES	_____
Upper/lower case keyboard input.	YES	_____
LO-Res graphics.	YES	_____
80 column display capability	YES	_____
Z-80 CP/M Ver. 2.x & Northstar also available	YES	_____
Affordable!	\$99.95	_____
Low cost enhancement option:		
Hi-Res turtle-graphics.	YES	_____
Floating-point mathematics.	YES	_____
Powerful package with own manual,		
50 functions in all,		
AM9511 compatible.		
FORTH-79 V.2 (requires 48K & 1 disk drive)		\$ 99.95
ENHANCEMENT PACKAGE FOR V.2		
Floating point & Hi-Res turtle-graphics		\$ 49.95
COMBINATION PACKAGE		\$139.95
(CA res. add 6% tax; COD accepted)		

MicroMotion

12077 Wilshire Blvd. #. 506
L.A., CA 90025 (213) 821-4340
Specify APPLE, CP/M or Northstar
Dealer inquiries invited.



Chances are, when you bought your first disk drive, it was an Apple. Now that you're ready for a second, take a look at Quentin.

Our Apple*-Mate™ 5¼" Disk Drive is fully software transparent with Apple's DOS 3.3 operating system in full and half track operation.

Add it to your present drive for greater capacity and faster access. Just plug it in and go to work.

And the Apple-Mate has these High Performance advantages:

ON TRACK HEAD SEEK

A precision lead screw positions the head onto the correct track. Time-consuming retries and disk-to-disk copying errors are virtually eliminated.

SIEMENS† DISK DRIVE

The apple-beige unit is built around the highly reliable

Siemens system with over 10,000 lifetime hours. Shielded connecting cable also attached.

LONG TERM DEPENDABILITY

MTBF (Mean Time Between Failures)—8,500 power-on hours, and the unit has a one-year warranty.

COUNT ON QUENTIN FOR QUALITY

Quentin Research was building disk systems for the computer industry when Apple was a little bud on the big computer tree. We're known for product reliability and stand behind every system we sell you.

But the best news may be the price—only \$335.00 (40 tracks).

A special introductory offer when you order Apple-Mate directly from us.

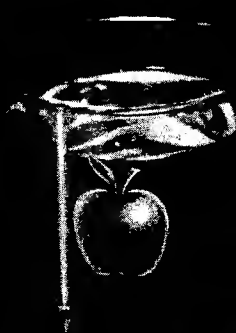
So when you're ready to boost the juice on your Apple, add-on the Quentin Apple-Mate.

To order: Check, money order, Visa or Mastercard number. Calif. residents add 6% sales tax. Allow one week delivery.

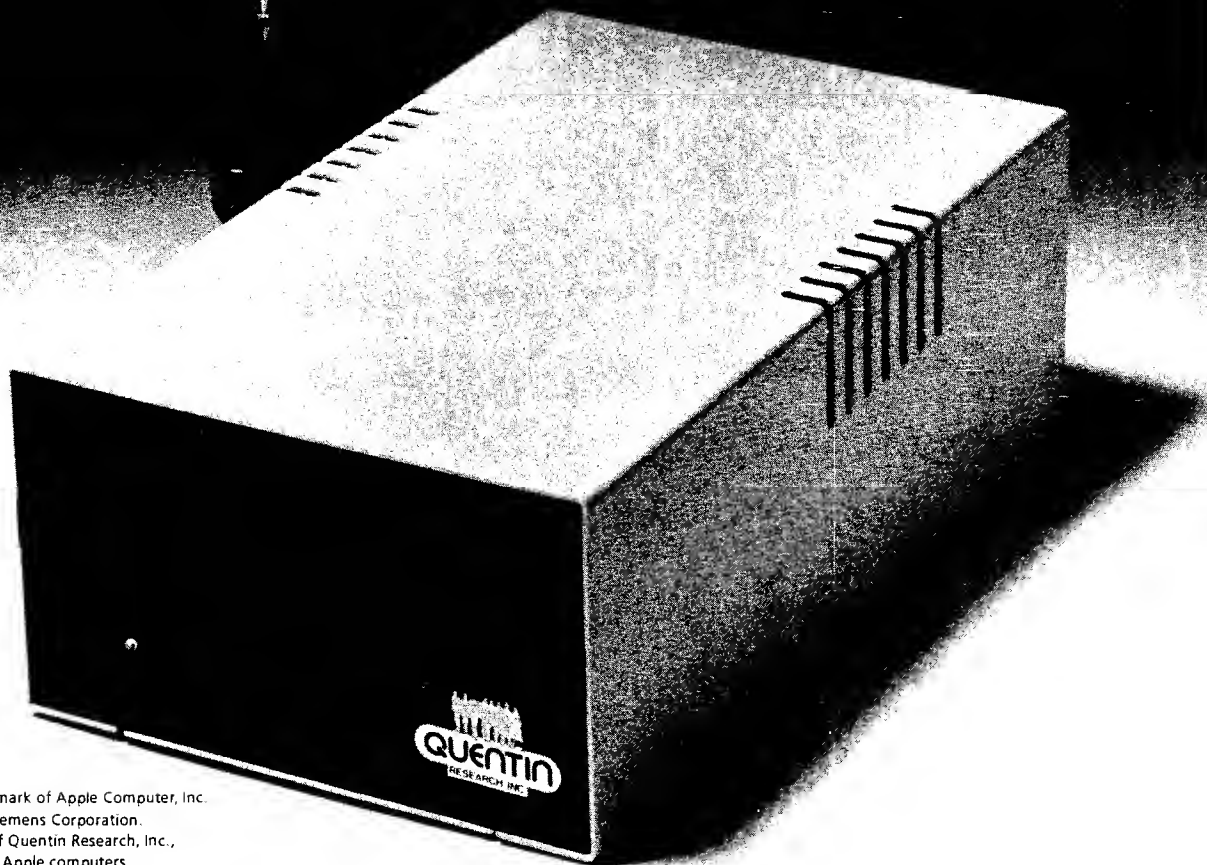


19355 Business Center Drive
Northridge, California 91324
(213) 701-1006

MORE JUICE FOR YOUR APPLE®



**Special
Introductory
Price:
\$335.00**



® Apple is a registered trademark of Apple Computer, Inc.

† Siemens is a trademark of Siemens Corporation.

* Apple-Mate is a trademark of Quentin Research, Inc., which does not manufacture Apple computers.

MICROTM

CoCo Bits

By John Steiner

This month's CoCo Bits re-examines the single disk COPY command. In addition, I have noted a few CoCo-related news items. One item I did not mention last month regards the transfer of machine-language files to disk. Before loading the routines into memory, be sure to reserve enough memory space so BASIC will not overwrite your program. Also, before loading and executing the modified BEDLAM from disk, a CLEAR 200, 16384 will protect the program from BASIC. Without this command, the program seems to execute properly but does not print the opening message.

As I mentioned last month, the single disk COPY command is available and will not destroy a program that is in memory (like DSKINI and BACKUP). This opens the door to a useful routine for selective backup of program and data files. The backup command is appropriate for archives and duplication purposes. COPY is useful when only a few files require transfer, or if program data must be transferred to a disk without destroying already existing files.

If several files must be transferred, however, it is tiresome to enter the files one by one using COPY "filename/ext". The program in listing 1 provides a selective backup routine. It reads the disk directory track and stores all the program names in a string array. The array holds up to 68 file names, the maximum number a CoCo disk can hold. After reading the filenames, each name is presented. Pressing "Y" invokes the COPY command and the file is read into memory. You are prompted to switch disks, and if all goes well, told that the copy is complete. If you don't wish to copy a file press any other key. The next file in line is then presented for your decision. Be sure to reinstall your source disk before pressing "Y".

In addition to the COPY command, the simple program makes use of another powerful disk command.

DSKIS\$ is used in a loop to read the sectors in the directory track. It is the only BASIC command that can directly read the directory. The routine that reads and stores the filenames is modified from the routine provided on page 62 of the COCO disk manual. By the way, there is a slight error in the routine that will cause it to miss several files. Line 60 reads FOR N=1 TO 7; it should read FOR N=0 TO 7.

The selective backup program routine uses several small arrays to read and identify the files that exist on a particular disk. Upon execution of line 160, the array FI\$ contains the filenames of the program on the disk. Lines 170 to 230 present the filenames and invoke the copy command if necessary. This routine has saved me a lot of time and hassle.

A Color Computer user's group has been formed in the Toronto, Ontario, Canada area. If you are interested in joining, you may contact Patricia Jackson at (416) 425-1116. Call weekdays after 6:00 p.m., or on the weekend. There is also a user's group in the Fargo, North Dakota area. Contact me and I will put your name on the meeting notice mailing list. Anyone

wishing to pass along similar information can contact me directly at the address shown below. It will take two to three months for your notice to appear in MICRO.

Rumors are that Tandy has signed an agreement with a group of RCA distributors to market the Color Computer in retail outlets not handling Radio Shack products. The new Color Computer will have a different color case and new name. If you have more details on this, or any other news regarding CoCo, pass it along.

Recently, I received an interesting musical program cassette. The classical rendition with four-voice organ music is the highest quality music routine I have heard, and I was impressed with the thought that most programmers are not using CoCo's sound abilities to their fullest. Several musical selections are available from Classical Software, 8931 Comanche Road, Longmont, Colorado 80501. They plan to announce a music editor with four-part tonal structure that will allow the user to enter and play notes directly from sheet music.

I own one of the early model Color Computers (serial number 337) and follow news about the Radio Shack 32K

Listing 1: COPY

```
10 CLS : PRINT@4, "SELECTIVE BACKUP PROGRAM"
20 PRINT@40, "BY JOHN STEINER"
30 PCLEAR 1
40 CLEAR 2000 : DIM FI$(67)
50 FOR X = 3 TO 11
60 DSKIS$ = 0, 17, X, A$, B$
70 C$ = A$ + LEFT$(B$, 127)
80 N$(0) = LEFT$(C$, 8)
90 EX$(0) = MID$(C$, 9, 3)
100 FOR N=0 TO 7
110 N$(N) = MID$(C$, N*32+1, 8)
120 EX$(N) = MID$(C$, 9+N*32, 3)
130 IF LEFT$(N$(N), 1) <> CHR$(0) AND LEFT$(N$(N), 1) <> CHR$(255)
    THEN FI$(K) = N$(N) + " / " + EX$(N) : K=K+1
140 NEXT N
150 NEXT X
160 CLS:PRINT@64, "ENTER Y TO COPY"
170 FOR J=0 TO K
180 PRINT@224, FI$(J)
190 Z$=INKEY$: IF Z$="" THEN 190
200 IF Z$="Y" THEN COPY FI$(J)
210 IF Z$="Y" THEN CLS : PRINT@224, FI$(J) " COPIED" : FOR I=1 TO 400
    : NEXT I
220 IF Z$="Y" THEN PRINT@0, "PLEASE REINSERT SOURCE DISK"
230 NEXT J
```

CoCo Bits *(continued)*

modifications. I have wanted to upgrade to the new version for a while, but have not wanted to be without CoCo for the time it would take to make the change. I did increase memory capacity by piggy-backing existing memory with 16K chips. It is a relatively inexpensive procedure and works well, giving fewer OM errors. One of the major disadvantages of this modification is that Radio Shack is replacing the early boards with an updated processor board and 64K RAM chips. The 64K chips are permanently wired making the upper 32K bank inaccessible. A few simple changes allow you to restore the upper bank and deselect the ROMs that normally reside there. The user can then load another DOS, modify BASIC, or change the entire character of CoCo. When Radio Shack changed the memory chips, the company had to issue a new Color BASIC ROM. Color BASIC 1.1, in addition to checking for and using 32K, has a few of the previous bugs removed. The 1.1 ROM will send 8-bit serial data

to the printer port. This allows CoCo to send graphics or special characters to the printer without loading Tandy's PTFX program.

I am interested in hearing from anyone who has modified a Color Computer to 64K without converting to the E board. I would also like to hear from FLEX and OS-9 users who successfully run their programs on CoCo. The added power and software compatibility is a major step for Color Computer programmers.

Next month, in addition to CoCo news, I will discuss some books available for Color Computer users. I will also take a look at medium- and high-resolution graphics modes available in Extended BASIC.

You may contact the author at 508 Fourth Avenue NW, Riverside, ND 58078.

MICRO™

TIRED OF TYPING? MICRO has the solution.

Order a diskette of three recent utility programs for the Apple. For only \$10.00, plus \$2.00 shipping and handling, you will receive a DOS 3.3 diskette containing the assembled listings of:

AppleSoft Variable Dump by Philippe Francois (MICRO, April 1982)

Straightforward Garbage Collection for the Apple by Cornelis Bongers (MICRO, August 1982)

COMPRESS by Barton Bauers (MICRO, October, 1982)

Please send check, money order, or VISA or MasterCard number. Only prepaid orders accepted. If you missed the above issues of MICRO they can be ordered now! Include \$2.50 for each issue.

Send orders to:

Apple Utility Disk
MICRO, P.O. Box 6502,
Chelmsford, MA 01824

FRANKLIN

ACE 1000

Apple II compatible
64K of RAM
Upper and lower case
Typewriter-style keyboard
12-key numeric pad
Alpha lock key
VisiCalc keys
50-watt power supply
Built-in fan

\$1199.00

MX-80FT

w/Graphics

\$544.00

AIM-65, 4K RAM \$465.00

COMPLETE CATALOG .. FREE


COMPUTERS PRINTERS MONITORS

MICROPROCESSOR COMPONENTS

Bedford Micro Systems

P.O. Box 1182, Bedford, Texas 76021

(817) 283-0013




A powerful utility that opens a window into the Color Computer's disks.

COLORZAP uses the power of the Color Computer to provide both rapid scanning and full screen modification capabilities. You can now examine, modify, and copy programs or data while they're stored on disk. Access them by filename or location.

COLORZAP is programmed largely in BASIC so that you can modify it if you'd like, but part of it is in machine language to provide fast response. All accesses to disk are performed with standard interfaces, so any standard Color Computer disk can be examined. You can directly access the disk's directory and control information to examine a clobbered disk, recover a killed file, or find parts of a file when other parts have been lost. With this new window into its disks, the Color Computer sheds its image as a toy. Now you can use this exciting machine like other powerful microcomputers.

For the TRS-80 Color Computer. Available on disk with an accompanying manual from **Software Options**, 19 Rector Street, New York, N.Y. 10006. 212-785-8285. **Toll-free order line: 800-221-1624.** Price: \$49.95 (plus \$3.00 per order shipping and handling). New York State residents add sales tax. Visa/Mastercard accepted.



**SOFTWARE
OPTIONS INC.™**

From Here to Atari

By Paul S. Swanson

Atari News

I was pleased to see that Atari, Inc., recently established two regional software acquisition centers located in Cambridge, Massachusetts and London, England. The centers were set up to acquire software by contracting out for specific programs, or by buying software that has already been developed independently. More centers are planned for the future; I'll let you know where they will be as soon as Atari announces that information.

Technical Tidbits

Code conversion is required in two areas when you're programming the Atari. The "normal" character code, called ATASCII, is a variation of ASCII. There are two other character codes used by the system. One is used to write characters to the screen. The screen handler does this conversion automatically when you PRINT to the screen, but if you use your own routines and put the characters directly on the screen with POKE or a similar method, you need to convert to this screen code.

The operating system manual includes a table that shows you the correspondence between ATASCII and the screen code (which they call the "Internal Code"). You can form a look-up table if you want by using a 256-byte string. Set it up so the value to POKE is the ASC(value of the byte in the string found at AVAL + 1, where AVAL is the ASC(value of the ATASCII character to be displayed.

An alternative approach, which consumes less memory than the look-up table, is using dependent IF statements. Using N as the ATASCII value to display:

```
FLAG = INT(N/128):N = N - FLAG * 64:
IF N > 95 THEN N = N - 96: IF N > 64
THEN N = N + 32
```

After you execute that one line of code (it must be in one program line),

POKE the screen location with N+FLAG. FLAG will equal 128 for inverse video characters and will equal zero for normal video characters in mode 0. There are two bits in modes 1 and 2 that determine the color, but the conversion routine in the above IF statements will interpret them both correctly.

The other code conversion would be for characters read from the keyboard. Several people have asked me how to eliminate the keyboard click. The only way to completely eliminate it would be to disconnect the keyboard speaker, but you can use another method if you write your programs to accommodate it. Instead of using INPUT and GET to obtain information from the keyboard, you can PEEK location 764. This location contains the keyboard code of the last key pressed on the keyboard. You must read this location, then POKE 764,255. If the location contains 255 you know that no key has been pressed since the last time you read it.

The problem with this method is that the code you read is neither ATASCII nor the internal code. You can get the values of all of these codes by running the following program:

```
10 REM ** KEYBOARD CODES **
11 REM ** STOP BY PRESSING BREAK **
12 REM **
13 REM **
20 PRINT "PRESS KEY AND THIS PROGRAM
30 PRINT "WILL DISPLAY THE
40 PRINT "CORRESPONDING KEYBOARD
   CODE AS A DECIMAL VALUE:"
50 N = PEEK(764)
60 IF N = 255 THEN 50
70 POKE 764,255
80 ? N;" ";
90 GOTO 50
```

If you use this program as a subroutine by itself, it will act as a GET statement. Putting the subroutine in a loop that stacks the codes in a string until it gets a RETURN code will act as an INPUT statement for alphanumeric

input. For this, remember to display the characters on the screen and to make allowances for backspaces. Now your program will not produce a click with each keystroke.

The only other common code conversions required are for the graphics screens. Those are simpler than the other conversions. If you are using the standard screen set up by BASIC, it is much easier to use standard BASIC statements like PLOT and DRAWTO. If you want to set up a specific shape that would require a lot of DRAWTO commands for a relatively small area, you may want to use PRINT.

Although converting to exact byte values to POKE onto the screen is possible, PRINT allows you to address each individual pixel on the screen. You PRINT an alphanumeric string to the screen through channel six. In mode 3, POSITION the graphics cursor at the beginning of one of the lines in the image, then PRINT #6;"112233" for two pixels each of colors 1, 2, and 3. To print the background color, which will allow you to erase an image, use zero, four, or a space. In two-color modes, use only zero and one. This method will save you substantial conversion over PEEKing and POKEing and will, in some cases, run much faster than the equivalent PLOT and DRAWTO statements. You don't need a COLOR statement for the PRINT method because you specify the color register directly, and there is an additional advantage to providing a version of the image right in the program (invaluable in debugging).

Next Month

My January column will introduce the Operating System and Hardware manuals and a few other sources of more technical information on the Atari. I plan to make the Technical Tidbits a regular feature, so send in your questions.

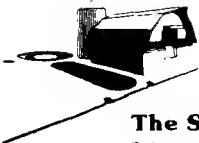
MONSTER MASH



It is late at night in a monster infested graveyard and you have been given the job of keeping the monsters in. All you have between you and complete chaos is a new MonsterMasher System and quick reflexes.

Monster Mash is an original and unique arcade action game written in assembly language for the Apple II and Apple /// (in emulation mode).

\$29.95



The Software Farm
3901 So. Elkhart
Aurora, CO. 80014
PH: (303) 690-7559



When all hell breaks loose.

Pandora's box is open. And all the evils
of the past are loosed upon mankind.
Armed with bolts of lightning, you
have the chance to recapture and
return these corrupt creatures
of doom to the prison of
Pandora's box. Time is short.
The world is already changing
for the worst. Your skills and
courage are needed.

PANDORA'S BOX

© DATA MOSI

9748 Cozyroff Ave., Chatsworth, Ca 91311 (714) 799-1202

VISA/MASTERCARD accepted. \$2.00 shipping/handling charge.
(California residents add 6.75% sales tax.)

Apple II is a trademark of Apple Computers, Inc.

MICRO

News

by Phil Daley, MICRO Staff Editor

Apple Bits and Pieces

As the release date for a new APPLE approaches, rumors fly fast and furious. Apple is securing sources for one million 68000 microprocessors, leading me to believe that the "Lisa" model (APPLE IV?) will be the first out, probably this Spring. It is to sell for approximately \$8000 and to be pitched at the business person who knows little about computers. At least, those are the rumors.

...

The "Seem alike" Franklin ACE 1000 may prompt Apple to release the Super Apple II sooner than originally anticipated. In addition to having 64K standard, rumor has it that the Super Apple II will contain far fewer chips on the mother board and will sell for substantially less.

The Franklin looks like an Apple II, especially when you take the cover off (the only noticeable difference is the larger power supply). The mother board looks almost identical, although somewhat enlarged. The chips are all the same and the I/O slots are similar. The Franklin is delivered with Applesoft and the Apple monitor ROMs installed. The other principal differences are that the Franklin accepts and displays lower case and has no color capabilities, soon to be remedied according to the manufacturer.

Having lost the preliminary injunction ruling against Franklin, Apple is asking for a reconsideration due to a similar case that ruled in favor of the manufacturer. Apple's position is that object code is copyrightable, and therefore proprietary and not usable by others.

Just to make the issue more complicated, Franklin is suing Apple for price manipulation and threatening Apple dealers who want to carry Franklin products.

Also pushing on the retail price are the Far East imitations, yet to be seen in the U.S., which are selling at one-fifth the normal European selling price.

...

There are rumors that the Mackintosh (also from Apple), a cheaper, simpler version of Lisa, is still in the developmental stage and is not expected until the end of next year at the earliest.

...

MICRO Bulletin Board

MICRO has instituted a sophisticated Bulletin Board/Information Service System on our Apple II, which will be available to subscribers Monday through Thursday nights from 5:00 PM to 8:00 AM Eastern Time. The MICRO Bulletin Board System is using software developed by

Computer Stations, Inc., of Granite City, IL, and a D.C. Hayes Associates, Inc., microcoupler. Our telephone number is (617) 256-1446.

After dialing into the Net-Works program, a self-explanatory menu is presented. The option (C)hat will not be supported. The first time that you log on you will be expected to leave your name, etc., for subscription verification.

This check will take at least one day. You will have only limited access to the system until your name has been verified and added to the queue of valid users. Please write down the password that the system assigns to you so that you can use it for future calls. A "<ctrl>S" will temporarily stop the system in case it is scrolling too fast to read. Generally, new users may read the system, but not write to the system until verified. We are planning a communications issue for April with articles on all aspects of computer communications. If you have written an article or have any suggestions or criticisms, please send them to us here at MICRO.

...

A Computer Center

A new resource center has been opened in Newton, MA, to meet the educational and instructional needs of executives who are interested in learning how to make effective use of desktop computers. Called The Computer Forum, this educational institution will offer integrated courses, software selection, continuing help, and customized seminars to interested individuals and businesses. Course offerings will include *How to Make Computers Work for You*, *Using Your Apple*, *Programming in BASIC*, *Data Bases*, *Using Business Graphics*, *The Electronic Spreadsheet*, *Advanced VisiCalc Techniques*, and *Management and Analysis Using VisiCalc*. The Forum has several classrooms, one for each system. Currently, only the Apple room is fully equipped, but plans call for an IBM PC room and possibly a XEROX room. Sign-up for the first schedule of courses has been brisk. We wish the Forum much success and hope that additional centers can be opened around the country.

MICRO

Statement of ownership, management, etc., required by the act of Congress of October 23, 1962, of MICRO, published monthly at Chelmsford, Massachusetts, for November 1982.

The name and address of the publisher is MICRO INK, 34 Chelmsford Street, Chelmsford, Massachusetts. The President/Editor-in-Chief is Robert M. Tripp of Chelmsford, Massachusetts.

The owner is THE COMPUTERIST, Chelmsford, Massachusetts and the names and addresses of stockholders owning or holding one percent or more of the total amount of stock are: Robert M. Tripp and Donna M. Tripp of Chelmsford, Massachusetts.

The known bondholders, mortgagees and other security holders owning one percent or more of the total amount of bonds, mortgages or other securities are: none.

The average number of copies of each issue of this publication sold or distributed through the mails or otherwise to paid subscribers during the twelve months preceding the date shown above is: 23,292.

I certify that the statements made by me above are correct and complete.

Signed: Robert M. Tripp
President/Editor-in-Chief

PIE WRITER.

Just compare it.

PEELINGS II Magazine did just that against 9 others. And PIE WRITER came out on top, rated AA+.

Why? Because PIE WRITER does everything a writer needs done, quickly, easily and inexpensively. If you can type, even with just two fingers, you'll be working comfortably with PIE within fifteen minutes.

Begin immediately typing text onto the screen. You can add or delete characters, words . . . sentences . . . paragraphs with one simple key stroke.

Misspellings? PIE WRITER's "search and replace" command corrects the word where you spot it . . . and then makes it correct throughout the entire text—be it a paragraph or a million dollar presentation.

When your document is finished, a touch of the key will print out 1 or 99 copies . . . in the format you want . . . neat, precise and always accurate.

You can start using PIE WRITER on your basic Apple II system the moment you get it home. Nothing extra to buy. But you'll never outgrow PIE because it works with just about anything you might want to add:

- 80-column boards
- shift-key modification
- lower case adapter
- spelling checker
- hard disks
- VisiCalc* files
- modems
- program source files

*VisiCalc is a registered trademark of VISICorp

Just how functional is PIE WRITER? Try these with any other word processor:

- Character, word, line and paragraph editing
- Global word search and replace
- Automatic centering
- Justify right and left plus incremental spacing
- "Help" screen for quick reference
- Cut and paste
- Automatic wrap around
- Forward and backward page scrolling
- Custom form letters and mailing lists
- Full format control
- Status display
- Full control over page numbering, bottom and top headers
- Save a segment of text on another file
- Insert text from another file
- Underlining
- Control page breaks
- Tabbing by word or moveable tab marks
- Built-in training lessons
- And Pie Writer's comprehensive, indexed manual and quick reference card make it easy to use; it is very writer-friendly

The best word processor available:
PIE WRITER. For ~~\$149.95~~
you just can't afford to buy anything else.

All you need to start is an Apple II with 48K, 3.3 DOS, one disk drive, a monitor and printer.

Available at your local computer store, or call

1-800-343-1218

(in MA, call 617/937-0200)

HAYDEN SOFTWARE

It's All Relative— CBM Disk Techniques, Part I

by Jim Strasma

Contributing editor Jim Strasma begins a series that explains how to get the most from CBM's powerful disk operating system. Examples are drawn from a well-written mailing list package that is both inexpensive and widely available. In Part 1 Jim covers global variables, combining BASIC with machine language, and chaining of program modules.

Editor's Note: To implement all of these techniques you should have a DOS 2.0 (or later) disk drive. BASIC 4.0 is also assumed. However, ways to emulate BASIC 4.0 disk commands from Upgrade BASIC and VIC BASIC are summarized.

One of the best features of Commodore's BASIC 4.0 and DOS 2 is its use of relative records for data files. This is a very powerful technique, not well matched by competing computers in Commodore's price range. However, relative records can be quite confusing, and though they have been around for two years now, are largely used in commercial programs. However, there is one large program package freely available that uses relative records — Chris Bennett's "Mail List 4040." In one form or another it has been around for about two years. For much of that time I have been modifying and documenting it.

With the help of the mail list, this series of six articles will thoroughly explain the use of relative records. It will also cover some programming techniques for large packages and a machine-language program that takes much of the drudgery out of data entry programming.

In this first article I will prepare the computer to run the mail list. In the

process, I will: 1. show how to mix BASIC and machine language, 2. have one program load another without stopping or losing variables (called *chaining*), and 3. explain the use of global variables (called *soft coding*).

Because of the general availability of Bennett's "Mail List," a full listing will not be presented here. However, you don't need the program to understand the articles. If you do wish to obtain the program, see the box on page 41.

Mixing BASIC and Machine Language

One of the more difficult tasks in programming is mixing BASIC and machine-language code gracefully. When first released, the mail list used one common method, reading the machine-language portion from data statements and POKEing it into working locations. This method easily allows changes to the BASIC program. However, if the machine-language portion is sizeable it can be slow; incorporating substantial changes from a new assembly of the machine-language portion would be tedious at best.

Next, I tried attaching the machine-language portion to the end of the BASIC code and using a machine-language SYS call to boot it into working location. This method is fast. However, it makes modifications to the BASIC program difficult, as any change in the length of the program also moves the machine code, guaranteeing a crash when the new version is used.

Now I use a small trick to load the machine-language portion separately from the BASIC part. This method is quick and allows easy changes to both the BASIC and machine-language portions of the program.

Line 1040 checks to see whether a key location contains the value it does when the machine code has been

loaded. If not, MEMSIZ, the zero-page location that controls top-of-memory pointers, is lowered along with FRETOP, the top-of-dynamic strings pointer. (On the VIC, MEMSIZ is at \$37 and FRETOP is at \$33.)

The two POKES protect the machine code from BASIC's dynamic string variables. Note that if only MEMSIZ were altered, BASIC would think it had a negative amount of memory free. Since changing these pointers ruins any variables already in the top of memory, it is essential to do it only at the beginning of the first program module.

```
1030 REM LOAD OBJECT PORTION
      IF HAVEN'T
1040 IF PEEK(31232) < > 76 THEN
      POKE 53,122:POKE 49,122
      :DLOAD "OBJECT CODE"
```

After resetting the memory pointers, line 1040 loads the machine-language portion from disk as a program named "object code." Usually loading a new program destroys the old one, but not this time. "Object code" loads very high in memory, beginning at location 31232, (\$7A00). It will overwrite anything else up there, such as Universal DOS support, but not BASIC programs located lower in memory.

Since the DLOAD command was part of a running program, BASIC attempts to execute "object code" as soon as it is fully loaded. However, BASIC assumes its programs begin where another pointer, TXTTAB points. In this case, we've left it alone. This means that BASIC will execute "mail list 4040" again. That is the main reason for checking to see whether "object code" has already been loaded. Otherwise we would never get past line 1040.

After the load the IF test in line 1040 fails and the program continues.

Chaining

Line 1060 is another line that must appear at the beginning of the first program module. For program chaining to work correctly, we must either make the first program the largest one, or else convince BASIC that this is so. We could do this by adding dozens of long lines to the program as ballast. However, this would add to its loading time, and take up more storage space on the disk. I have only followed that idea to the extent of coding this module very loosely, with mostly single-statement lines and lots of REMark statements. The added clarity is worth the slight waste. I also started with line number 1000 to keep all line numbers the same length, again for clarity.

In early versions of the mail list, chaining worked by altering the file size pointer, VARTAB at location 42 (\$2A), as each module began. This worked because BASIC keeps track of the actual file size in pointer EAL, at location 201 (\$C9), during a load. (On VIC, VARTAB is at \$2D and EAL is at \$AE.) We simply had a line like the one below at the start of each module.

```
10 POKE 42,PEEK(201):POKE 43,
    PEEK(202):CLR
```

Unfortunately, it won't work without the CLR, and once CLR is used, the old variables are gone. This means that a separate disk file has to be established and loaded by each module to remember global variables, or the variables have to be hidden from BASIC and PEEKed. Either method is slow.

By POKEing VARTAB with a value at least as large as it would need to run the largest module, we can use line 1060 instead of line 10, and need it only in the first module.

```
1060 POKE 42,0:POKE 43,53:CLR
```

To determine the correct values to use here, load the longest module in your program, and enter:

```
?PEEK(43)
```

Add two to the result and write it down. Use that number in place of 53 in line 1060. Note that we could have also PEEKed at 42, but I prefer to overstate slightly the required memory. This allows minor additions to that longest module without also requiring a change here.

Don't make program changes to any module after loading it via a chain. BASIC no longer knows the module's true size. Instead, reload the module from disk in immediate mode and then make the changes. This is especially important if you have used line 10 above. EAL isn't changed by line editing. If EAL points lower than the end of a modified BASIC program, line 10 would force the variables to begin being stored on top of the last lines, ruining them. To prevent such disasters, it's always a good idea to save a modified program to disk before trying to run it.

The actual chaining happens in line 2060:

```
2060 DLOAD D(PD), "4040 MENU"
    ON U(UN)
```

For BASIC 2.0 and the VIC use:

```
2060 LOAD STR$(PD) + ":4040
    MENU",UN
```

Soft Coding

Notice the variables used in line 2060 above: PD and UN (program drive and disk unit number). They are set earlier in the program, in lines 1220 and 1230:

```
1220 UN=8:REM DISK UNIT
1230 PD=0:REM PROGRAM DRIVE
```

By setting them there and using only the variable names everywhere else in the program package, it is easy to change the package to work with different equipment, such as a disk drive that answers to device 9 instead of 8. We will have more to say about soft coding shortly, but first we need to finish setting up.

Setting Text Mode

One other task awaits us in preparing the machine. Commodore computers have two character sets, one for graphics and one for upper- and lower-case text. Since this program uses text, we must enable the text character set. A method that works for all CBM and PET models is given in lines 1080 and 1090 below. (On the VIC, leave out line 1080.)

```
1070 REM SET TEXT MODE
1080 POKE 59468,14
1090 IF PEEK(57345) < > 54 THEN
    PRINT CHR$(14):REM UNLESS
    FAT 40
```

For reasons that make sense only to Commodore, Fat 40's, (the 4016 and 4032 with 12" monitor), are adjusted on the assembly line so that printing CHR\$(14) zooms the top and bottom lines off the screen. The IF test in line 1090 prevents this. However, there is also a hardware fix. On the underside of the video display board is a hole labeled "height." Your dealer can adjust your display in about 30 seconds to restore the lost top and bottom lines permanently. If you do it yourself, remember that metal screwdrivers are good conductors and the video board carries 10,000 volts. One slip could do more than violate your warranty.

The CHR\$(14) is especially needed by 80-column models. If you leave it out and the machine was previously in graphic mode, lines will appear squished together.

The matching lines to enable the graphic character set are:

```
1070 REM SET GRAPHIC MODE
1080 POKE 59468,12
1090 PRINT CHR$(142)
```

Leaving out the CHR\$(142) on 80-column models leaves them with a venetian blind effect, separating lines of graphic characters. No Fat 40 fix is needed this time. (Line 1080 should still be omitted on the VIC.)

Always establish one character set or the other at the start of any program package. CBM models start up in text mode, but PET models start in graphic mode.

Initialization

At this point the machine is ready. The machine-language portion is in and protected. The file pointers have been set for successful chaining and the character set is correct. Now the program begins a long process of initializing variables. Because this takes about five seconds, it is wise to give the user something to look at meanwhile. The mail list starts with a copyright message and then a status line:

```
1200 PRINT "          INITIALIZING
```

This assures the user that the program hasn't died. If the delay will be more than half a minute, also give the user an estimate as to how long the task should take and an occasional progress report.

More on Soft Coding

In the lines following 200 in this first module, the global variables are defined. Because they are not cleared by later modules, the way the entire package works can be modified drastically by changing a single line in this module. Naturally, the other modules have to be carefully written to take advantage of this power. We will see how this is done later in this series of articles.

The global variables used tend to fall into three categories: those that define messages, those that define special characters, and those that act as flags to control the program. The first category allows easy changes to such things as field names or default field contents. These messages may also include cursor control characters to be sure they appear at the correct location on the screen. To ease this task, the mail list predefines a position string of cursor controls in line 1880:

```
1880 POS$ = "[HOME,23DOWN,
      7RIGHT]" + " "
```

The characters shown in square

brackets represent literal cursor characters. The codes stand for one home character, followed by 23 cursor downs, followed by seven cursor rights. In the actual mail list, the literal characters are used and the codes are in a REMark statement at the end of the line. Always try to explain lengthy strings made up of cursor controls, especially if anyone will ever need to list your program to a non-Commodore printer.

Later lines select needed portions of the program with LEFT\$, as in line 1940:

```
1940 M2$ = LEFT$(POS$,8) + "START
      POSITION :"
```

However, we must be sure the messages are stored in high memory where they will chain correctly. To do this, we concatenate a null string to each literal string in the program, as shown at the end of line 1880.

If we didn't add the null string, BASIC would save space by pointing variable POS\$ at its original memory location in line 1880. After chaining, this location would likely contain

something quite different, and the string would be ruined. Adding the null string forces it into high memory where it is safe.

The second category of variables is illustrated by line 1830:

```
1830 QT$ = CHR$(34)
```

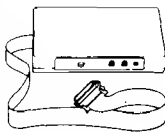
This is the quote character. It is needed later to allow INPUT# statements to read past troublesome characters like commas. We could use CHR\$(34) everywhere instead, but CHR\$ is a slow command in BASIC. Predefining QT\$ is at least ten times faster overall. Other characters the mail list predefines include RETURN, SHIFTED-RETURN, and SHIFTED SPACE. We will explain how each is used later in this series of articles.

The third class of global variables is the controllers. These include both numeric and string variables, used in IF tests and within expressions later in the program. For instance, line 1210 flags whether or not you want to allow the user to get out of the program by pressing STOP:

(continued)

SIGNALMAN MARK I DIRECT CONNECT MODEM - \$89.50

Standard 300-baud, full duplex, answer/originate. Powered by long lasting 9-volt battery (not included). Cable and RS-232 connector included.



EPROMS - HIGH QUALITY, NOT JUNK

Use with PET, APPLE, ATARI, SYM, AIM, etc. 450 ns. \$6.50 for 2716, \$12.50 for 2532. We sell EPROM programmers for PET and ATARI

5 1/4 INCH SOFT SECTORED DISKETTES

Highest quality. We use them on our PETs, APPLES, ATARIs, and other computers. \$22.50/10 or \$44.50/20



NEW! C. ITOH STARWRITER F-10 DAISY WHEEL PRINTER

Letter quality, flawless copy at 40 char/sec. Bidirectional printing. 15-inch carriage, uses standard Diablo ribbons and print wheels. ~~IEEE = \$1595~~

PARALLEL - \$1495, RS-232 - \$1680, TRACTORS - \$210

MAE SOFTWARE DEVELOPMENT SYSTEM FOR PET, APPLE, ATARI

"The Compatible Assembler"

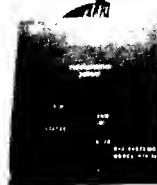
- Professional system for development of Machine Language Programs. 31 Characters per label.
- Macro Assembler/Text Editor for Disk-based systems.
- Includes Word Processor for preparation of Manuals, etc.
- Standard Mnemonics - Ex.: LOA (LABEL), Y
- Conditional Assembly, Interactive Assembly.
- Editor has string search/search and replace, auto line numbering, move, copy, delete, u/c capability.
- Relocating Loader to relocate object modules.
- Designed with Human Factors Considerations.

\$169.95

FLASH!! EHS Management has decided to allow \$50.00 credit to ASM/TED owners who want to upgrade to MAE. To get this credit, return ASM/TED manual with order for MAE.

ATARI AND PET EPROM PROGRAMMER

Programs 2716 and 2532 EPROMs. Includes hardware and software. PET = \$75.00 - ATARI (includes sophisticated machine language monitor) = \$119.95



PET BASIC SCROLL PROGRAM

Scroll thru basic program using Cursor up/down keys. Specify computer. \$6.00 on cassette, \$9.00 on disk.

Flip 'N' File diskette storage case (50-60 disks) - \$21.95

Memory Test for Apple on Disk = \$9.95, on Tape = \$6.95

System Saver for Apple - Fan, Surge Protection, 2 extra outlets, Apple power cord = \$75.00

BMC Green Screen Video Monitor.

12 inch CRT, sharp, crisp 40 or 80 column display. = \$90.00

DC Hayes Smart Modem = \$235.00, Micro Modem II = \$289.00, Chronograph = \$225.00

C. Itoh Prowriter Printer. Better than MX80. We use constantly with our Apple and PET. Can be used on IBM, Atari, TRS-80, etc. 120 cps, friction and tractor feeds, hi resolution dot graphics, nice looking, high quality construction. Parallel - \$499.00, with IEEE interface for commodore - \$599.00, RS232 - \$660.00

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!



1210 NS=0:REM NON-STOP?

If NS=1, the program becomes non-stop; a great idea when using untrained operators, but a terrible idea when a skilled user is trying to modify the package.

An example of a string variable used as a control is PZ\$, defined in line 1310:

1310 PZ\$="A":REM ASCII, P=PET

One of the skills of the machine-language portion of the package is that it can convert strings from PET ASCII to true ASCII codes and back again. This is useful when working with a modem or a non-Commodore printer. Line 1760 shows how this feature is used or skipped, depending on the contents of PZ\$:

1750 REM FLIP CASE OF ASCII
PRINTER PROMPTS
1760 IF PZ\$ < > "A" THEN 1830
1770 SYS SM,1,NA\$
1780 C3\$=C1\$
1790 SYS SM,2,C3\$

1800 C4\$=C2\$
1810 SYS SM,2,C4\$

My personal copy of the mail list carries the control variable idea a step further by using the variable TY to select between using the package as a church mail list, a computer users' mail list, and a sermon file, depending on whether TY=1, 2, or 3 in a new line added to this module.

The other special options set by the global variables are explained in the instructions that come with the mail list package, so I won't take space for them here. However, if you do get the program, notice that all the simple variables are defined before the arrays are defined. Doing things in this order cuts the initialization delay by 2.5 seconds. Further speed gains are possible by arranging the lines so the most-used variables and arrays are defined before those used less often. The ones most heavily used are usually inside nested loops and often-used subroutines.

Using Program Intelligence

The program selects either an ASCII

or a PET printer, as we saw in line 1310. However, it doesn't simply assume the printer is on, but goes to the trouble of checking, in lines 1350-1380:

1300 DV=4:REM PRINTER
1340 REM BE SURE PRINTER IS ON
1350 OPEN 4,DV
1360 PRINT#4,CHR\$(7):REM BELL
1370 IF ST THEN PZ\$="N":
PRINT " PRINTER IS OFF
1380 CLOSE 4

Line 1360 tries to print a BELL character to the selected printer device. If it succeeds, the IF test of the status variable will fail in line 1370. Otherwise, a warning is printed and the printer control variable is set to show no printer is on line. This allows users without a printer to safely use the package.

A similar technique is used in lines 1250-1290:

(continued)

PET / CBM™ SOFTWARE SELECT!

8032 OR **4032**
DISPLAY **DISPLAY**

FROM THE KEYBOARD OR PROGRAM
NOW RUN WORD PRO 3 OR WORD PRO 4

FROM THE SAME MACHINE

Available for either 4000 or 8000 Series

ALSO:

For **2001 / 3000** Series Computers

Operate these Models in a Full **8032** Like
Display For Word Pro 4*
and all other 80 Column Software
All installation instructions included.

EXECOM CORP.

1901 Polaris Ave.
Racine, WI 53404
Ph. 414-632-1004

PET/CBM a trademark of Commodore Business Machines
*trademark of Professional Software, Inc.

MICRO SPEC

Quit Playing Games . . .

Disk Based Software to Make Your
Computer Get Down to Business

Disk Based Data Manager—Create and manage your own data base. Allows you to create, add, change, delete, search, sort, print, etc. Available for VIC-20, Commodore 64, any CBM or Pet, and IBM Personal Computer.
VIC-20 59.95 All others 79.95

Inventory Control Manager—Fast, efficient inventory package which will manage your day to day inventory requirements. Provides information on sales and movement of items.

Mailing List Manager—4,050 items per 8050 disk, 1,300 on 4040 disk and 1,200 on 1540/1541 disk. User defined label format (1-4) across.

Payroll System—Full featured complete Payroll System. Up to 350 employees on a 8050 disk. Prints checks, 941's and W-2's. For the CBM 8032/8050, 4032/4040, Commodore 64/1541.

Hospitality Payroll—The most complete payroll system written specifically for the Restaurant Industry available today. Recognizes tip and meal credits, pay advances, salaried and hourly employees, etc. For the CBM 8032/8050.

CONTACT US FOR ALL YOUR DISK BASED SOFTWARE NEEDS

Call for specifics on Hardware Configurations.
Send Self-Addressed Stamped Envelope for
Catalogue of Games and other Applications
DEALER INQUIRIES WELCOME



2905 Ports O'Call Court
Plano, Texas 75075
(214) 867-1333



VISA and MASTERCARD Accepted

```
1240 REM SELECTS DATA DRIVE
1250 DD = 1
1260 OPEN 15,UN,15
1270 PRINT#15,"INITIALIZE" +
    STR$(DD)
1280 IF DS = 74 THEN DD = 0:REM IF
    SINGLE DRIVE
1290 CLOSE 15
```

As these lines initialize disk drive one, they identify single drive units and prepare the program to work with either single or dual drives.

An earlier version of the program had the user select one or two drives manually by changing line 250. However, I use both single and dual drives often, and decided it made more sense to let the computer use its own intelligence to work with all Commodore disk drives. This kind of intelligence in a program means more work for the programmer once, but less work for all the users for years to come. Programs you expect to give or sell to others should work on all existing and likely models. (If I followed that advice fully, this program would have used BASIC 2.0 disk commands, at some cost in

speed and a great cost in clarity.)

Next time we will begin working with relative records — creating the files needed by the mail list package.

MICRO

How to Obtain Bennett's "Mail List"

Many users' groups will have this program in their libraries. It is also available from ATUG (200 S. Century, Rantoul, IL 61866), TPUG (381 Laurence Ave W., Toronto, Ontario M5M 1B9, Canada), or from the author as part of his HELP disk. The HELP disk is a companion to the third edition of Osborne/McGraw-Hill's *CBM and PET Computer Guide* (edited by the author).

To obtain the HELP disk send \$15 to the address below. Specify 4040/2031 or 8050 format.

HELP Disk
Jim Strasma
1280 Richland Ave.
Lincoln, IL 62656

WHAT'S WHERE IN THE APPLE A Complete Guide to the Apple Computer

This **REVISED EDITION** of the famous Apple Atlas provides Apple computerists with a framework for understanding both the overall organization and structure of the Apple system and programming techniques that exploit that knowledge.

What's Where in the Apple contains the most complete memory map ever published as well as detailed information needed for actual programming.

All for only \$24.95
(plus \$5.00 s/h)

For owners of the original edition, MICRO is offering a companion book, *THE GUIDE to What's Where in the Apple*, for only \$9.95 (plus \$2.00 s/h).

THE GUIDE contains all new material that explains and demonstrates how to use the atlas and gazetteer published in the original volume of *What's Where in the Apple*.

VISA and MasterCard accepted

What's Where in the APPLE



A Guide to
THE APPLE COMPUTER

MICRO makes it easy to order!
Send check to:

MICRO INK
P.O. Box 6502
Chelmsford, MA 01824

Or call our toll-free number:

1-800-345-8112
(In PA, 1-800-662-2444)

MA residents add 5% sales tax

STATISTICS

PURE AND SIMPLE



Human Systems Dynamics programs offer you flexibility, accuracy, and ease of use. You can purchase from the HSD statistics specialists with complete confidence. Any program that doesn't suit your needs can be returned within 10 days for full refund.

NEW

STATS PLUS

\$200.00

Complete General Statistics Package
Research Data Base Management
Design and Restructure Your Files
Count, Search, Sort, Review/Edit
Add, Delete, Merge Files
Compute Data Fields, Create Subfiles
Interface with other HSD programs
Produce Hi Res bargraphs, plots
1-5 way Crosstabulation
Descriptive Statistics for all Fields
Chi-Square, Fisher Exact, Signed Ranks
Mann-Whitney, Kruskal-Wallis, Rank Sum
Friedman Anova by Ranks
10 Data Transformations
Frequency Distribution
Correlation Matrix, 2 way Anova
r, Rho, Tau, Partial Correlation
3 Variable Regression, 3 t-Tests

ANOVA II

\$150.00

Complete Analysis of Variance Package
Analysis of Covariance, Randomized Designs
Repeated measures Designs, Split Plot Designs
1 to 5 Factors, 2 to 12 Levels Per Factor
Equal N or Unequal N, Anova Table
Descriptive Statistics, Marginal Means
Cell Sums of Squares, Data File Creation
Data Review/Edit, Data Transformations
File Combinations, All Interactions Tested
High Resolution Mean Plots, Bargraphs

HSD REGRESS

\$99.95

Complete Multiple Regression Analysis
Up to 25 Variables, 300 Cases/Variable
Correlation Matrices, Descriptive Statistics
Predicted & Residual Scores, File Creation
Regression on Any Subset of Variables
Regression on Any Order of Variables
Hi-Res Scatterplot & Residual Plot
Keyboard or Disk Data Input
Case x Case Variable x Variable Input

Apple II, 48K 1 or 2 Disk Drives
3.3. DOS, ROM Applesoft

Call (213) 993-8536 to Order

or Write:

HUMAN SYSTEMS DYNAMICS
9249 Reseda Blvd., Suite 107
Northridge, CA 91324

VISA

master charge

Squeeze for PET BASIC Program

by Hans Hoogstraat

This short routine removes the unnecessary spaces, REMs, and blank lines from a BASIC program. It is relocatable and does not require maintaining two versions of the BASIC program.

SQUEEZE

requires:

PET/CBM — original, upgrade, or 4.0 ROMs

This routine squeezes all the imbedded blanks, line separators, and comments from a BASIC program. In addition, the following syntax corrections are made:

1. GO TO = GOTO
2. IF GOTO = IF .. THEN
3. IF .. THEN GOTO = IF .. THEN

SQUEEZE is relocatable and can be stored in either cassette buffer. It is designed to be called with a SYS command in the first line of your BASIC program. This means that you need to store only one copy — fully commented and expanded — of your program on tape or disk. When you run the program, it is automatically compressed first.

BASIC Example Program:

[XXX = ADDRESS OF SQUEEZE ROUTINE]

```
10 SYSXXX
15 :
20 REM EXAMPLE PROGRAM
25 :
30 PRINT "EXAMPLE PROGRAM"
35 :
40 FOR I = 1 TO 10
45 ::PRINT I, SQR(I)::REM ROOTS
50 NEXT
55 :
60 IF I < > 0 THEN TO TO 80 ::
```

```
65 :
70 I = 1::B = 1: REM NONSENSE
75 :
80 END
```

After the SYSXXX squeeze call, the program continues execution with the following BASIC code:

```
10 SYSXXX
30 PRINT"EXAMPLE PROGRAM"
40 FORI=1TO10
45 PRINTI,SQR(I)
50 NEXT
60 IFI < > 0THEN80
70 I=1:B=1
80 END
```

Cautions:

1. Do not use SYS XXX; any blanks between SYS and XXX can confuse the BASIC run-time pointers.
2. Any GOTO, GOSUB, or THEN references to REM-commented lines or : null lines will become erroneous due to the deletion of these lines. [Ed. note: SQUEEZE does not handle these references.]

SQUEEZE can be loaded into the first or second cassette buffer and can then be permanently saved with the BASIC program using the machine-language monitor SAVE command, or it can be made part of the program with DATA statements containing the machine-language code to be transferred to a suitable spot in memory using POKE commands.

Here is the procedure to save a BASIC program with SQUEEZE in the cassette buffer. [Original ROM: use first cassette buffer — \$027A - \$0339; upgrade ROM: use either cassette buffer — \$027A - \$0339 or \$033A - \$03F9; 4.0 ROM: use second cassette buffer — \$033A - \$03F9.]

1. Load SQUEEZE routine into correct buffer.
2. Type NEW and load BASIC program.

3. Type SYS4, which will display (4.0 ROM)

```
PC IRQ SR AC XR YR SP
.; 0005 E455 30 00 5E 04 F0
```

4. Type .M 002A 002B to display the start-of-BASIC variables pointer, which is usually the same as the end-of-BASIC text pointer. Assume the following display from the above command:

```
.M 002A 002B
.; 002A 4B 04 4B 04 4B 04 00 80
```

5. Now, to save the BASIC program and the SQUEEZE routine together on disk assuming SQUEEZE was loaded in the first cassette buffer, type

```
.S "0:EXAMPLE",08,027A,044B
```

027A = Start address of first cassette buffer.

044B = Contents of end-of-BASIC text pointer as displayed in locations \$002B-\$002A.

For tape use 01 instead of 08.

General Information

All CBM system labels references are consistent with the labels specified in Appendix F of the *PET/CBM Personal Computer Guide* by A. Osborne.

Hexadecimal dumps of the routine assembled for the three different versions of the PET ROMs are included in this article.

With some minor pointer modifications, the SQUEEZE routine should also operate on most other 6502 systems.

Hans Hoogstraat is a scientific research and systems development software and hardware consultant to the petroleum industry. You may contact him at Box 20, Site 7, SS 1, Calgary, Alberta, Canada T2M 4N3.

Listing 1: SQUEEZE Assembled for 4.0 ROMs

```

0010 ;SYSTEM EQUATES
0020 ;
0030 ;BASIC .DI 1 ;ORIGINAL ROM
0040 ;BASIC .DI 3 ;UPGRADE ROM
0050 ;BASIC .DI 4 ;BASIC 4.0
0060 ;
0070 BASIC .DI 4
0080 ;
0090 ;-----
0100 ;----- SQUEEZE -----
0110 ;-----
0120 ;
0130 ;THIS ROUTINE SQUEEZES A BASIC PROGRAM FROM ALL ITS
0140 ;IMBEDDED BLANKS, LINE SEPARATORS AND COMMENTS.
0150 ;
0160 ;IN ADDITION THE FOLLOWING SYNTAX CORRECTIONS ARE MADE:
0170 ;
0180 ;1. GO TO ..... = GOTO
0190 ;2. IF ..... GOTO = IF .. THEN
0200 ;3. IF .. THEN GOTO = IF .. THEN
0210 ;
0220 ;BASIC REFERENCES.
0230 ;
0240 ;           IFE BASIC-1
0250 BPOINT .DI $7A
0260 WORK .DI $A6
0270 LNKPRG .DE $C430
0280 ;
0290 ;
0300 ;           IFE BASIC-3
0310 BPOINT .DI $23
0320 WORK .DI $54
0330 LNKPRG .DE $C442
0340 ;
0350 ;
0360 ;           IFE BASIC-4
0370 BPOINT .DI $28
0380 WORK .DI $54
0390 LNKPRG .DE $B4B6
0400 ;
0410 ;
0420 ;           .BA BPOINT
0430 ;
0440 TXTTAB .DS 2 ;POINTER TO START OF BASIC
0450 VARTAB .DS 2 ;POINTER TO START OF VAR.
0460 ARYTAB .DS 2 ;PNTR TO START OF ARRAY TA
0470 STREND .DS 2 ;POINTER TO END OF VAR.
0480 ;
0490 ;PAGE ZERO WORK AREAS.
0500 ;
0510 ;           .BA WORK
0520 ;
0530 INPPTR .DS 2 ;INPUT LINE POINTER.
0540 NXTLIN .DS 2 ;NEXT BASIC LINE ADDRESS
0550 OUTPTR .DI VARTAB ;OUTPUT LINE POINTER.
0560 INPIND .DS 1 ;INPUT TEXT INDEX.
0570 OUTIND .DS 1 ;OUTPUT TEXT INDEX.
0580 OUTSEG .DS 1 ;OUTPUT LINE SEGMENT LENGT
0590 QTFLAG .DS 1 ;QUOT FOUND FLAG.
0600 PRVOUT .DS 1 ;PREVIOUS OUTPUT CHARACTER
0610 IFFLAG .DS 1 ;IF TOKEN FOUND FLAG.
0620 ;
0630 RAMLOC .DI $400 ;START BASIC TEXT
0640 ;
0650 ;BASIC TOKEN EQUATIONS.
0660 ;
0670 GOTOK .DI $89 ;GO TO
0680 IFTK .DI $8B ;IF
0690 REMTK .DI $8F ;REM
0700 TOTK .DI $A4 ;TO
0710 THENK .DI $A7 ;THEN
0720 GOTK .DI $CB ;GO
0730 ;
0740 ;-----
0750 ;
0760 ;           .BA $33A
0770 ;
0780 ;SET BASIC OUTPUT LINE ADDRESS POINTER.
0790 ;
0800 SQUEEZE LDA #L,RAMLOC+1
0810 STA #OUTPTR
0820 LDY #H,RAMLOC+1
0830 STY #OUTPTR+1
0840 ;
0850 ;SET BASIC INPUT LINE ADDRESS POINTER.
0860 ;
0870 NEXTLIN STA #INPPTR
0880 STY #INPPTR+1
0890 ;
0900 ;RESET ALL BASIC SCAN LINE FLAGS.
0910 ;
0920 LDY #0
0930 LDX #0
0940 ;
0950 ;COPY BASIC LINK AND LINE NUMBER FROM INPUT TO OUTPUT.
0960 ;
0970 COPYLNK LDA <INPPTR>,Y
0980 STA <OUTPTR>+Y
0990 STA NXTLIN,Y

```

(Continued on next page)

TURN AN EPSON PRINTER INTO A DAISY...

with the
SUPER-MX CARD
for the APPLE II.

The standard of printing excellence is the daisy-wheel printer. The SUPER-MX card provides the Epson printers with just about the same quality print as the daisy-wheels!

SUPER-MX Roman font is the standard.

Epsons can now print Elite with the SUPER-MX card.

Other optional font styles are available in addition to the standard Roman font that just plug into the extra sockets provided. They come in pairs so you can add a total of four extra fonts. Orator Large comes with Letter Gothic. Script comes with Olde English.

LETTER GOTHIC is modern looking.

ORATOR is easy to read and good for speeches.

SCRIPT adds the personal touch.

OLDE ENGLISH is very formal and elegant.



Apple Hi-Res graphics is fully supported with a wide variety of commands including: double dumps (both pages side by side), dump from page 1 or 2, double size, emphasized, rotated, strip chart recorder mode, and text screen dump.

The two expansion sockets allow EPROM expansion to 12K to insure you that the SUPER-MX card will remain the most intelligent interface around.

An Epson MX-80 needs Grafrax or Grafrax-Plus. An MX-100 requires Grafrax-Plus. Warranty is 90 days.

SUPER-MX card with cable \$175.00
Orator and Letter Gothic Fonts .. \$30.00
Script and Olde English..... \$30.00

Cash, cashiers check or money order. Personal checks will require 2 weeks to clear. California residents add 6½% sales tax.

Sples Laboratories
(pronounced "speez")
P.O. Box 336
Lawndale, CA 90260
(213) 644-0056

Apple II is a TM of Apple Computer, Inc.
Grafrax is a TM of Epson America, Inc.

Announcing THE GUIDE

A Complete Guide
to the Apple Computer



If You Own the Original
**What's Where in the
APPLE?**

You Will Want
THE GUIDE
A Complete Guide
to the Apple Computer

only \$9.95*

The Guide provides full explanatory text to lead you through the most complete Apple memory map ever published!

The Guide explains and demonstrates how to use the atlas and gazeteer published in the original volume!

If you missed the first edition of **What's Where in the Apple?**, a new revised edition containing **BOTH** the original atlas and gazeteer **AND** the all new Guide is available in one 256-page, Wire-O-Bound book for only **\$24.95!**

*MICRO makes it easy to order:
Send check (payable to MICRO) to:*

MICRO INK
P.O. Box 6502
Chelmsford, MA 01824

Call our toll-free number:

1-800-345-8112
(In PA, 1-800-662-2444)

VISA and MasterCard accepted

*Add \$2.00 shipping per book.

MA residents add 5%.

83-370

Listing 1 (continued)

```

0351- 96 5A      1000      STX #OUTSEG,Y
0353- C8         1010      INY -
0354- C0 04      1020      CPY #4
0356- 90 F2      1030      BCC COPYLNK
                  1040      ;
                  1050      ;--- CARRY SET ---.
                  1060      ;
                  1070      ;SET START BASIC INPUT AND OUTPUT TEXT INDEXES.
                  1080      ;
0358- 84 58      1090      STY #INPIND
035A- 84 59      1100      STY #OUTIND
                  1110      ;
                  1120      ;CHECK FOR END OF BASIC TEXT.
                  1130      ;
035C- A0 01      1140      LDY #1
035E- B1 2A      1150      LDA (OUTPTR),Y
0360- D0 16      1160      BNE SCAN
                  1170      ;
                  1180      ;ADJUST START OF VARIABLE ADDRESS.
                  1190      ;
0362- A2 05      1200      LDX #5
                  1210      ;
0364- A4 2B      1220      LDY #VARTAB+1
0366- A5 2A      1230      LDA #VARTAB
0368- 69 01      1240      ADC #1
036A- 90 01      1250      BCC CLR
036C- C8         1260      INY -
                  1270      ;
                  1280      ;PERFORM BASIC CLR
                  1290      ;
036D- 94 2A      1300      CLR STY #OUTPTR,X
036F- CA         1310      DEX -
0370- 95 2A      1320      STA #OUTPTR,X
0372- CA         1330      DEX -
0373- 10 F8      1340      BPL CLR
                  1350      ;
                  1360      ;FIX BASIC LINKS AND RETURN TO CALLER.
                  1370      ;
0375- 4C B6 B4   1380      LINK JMP LNKPRG
                  1390      ;
                  1400      ;-----
                  1410      ;
                  1420      ;SCAN BASIC INPUT TEXT LINE.
                  1430      ;
0378- A4 58      1440      SCAN LDY #INPIND
037A- B1 54      1450      LDA (INPTR),Y
037C- E6 58      1460      INC #INPIND
                  1470      ;
037E- A6 58      1480      LDX #QTFLEG
0380- D0 45      1490      BNE OUTTEXT
                  1500      ;
0382- C9 20      1510      CMP #'
0384- F0 F2      1520      BEQ SCAN
                  1530      ;
0386- C9 8F      1540      CMP #REMTK
0388- D0 01      1550      BNE CKSEG
                  1560      ;
038A- 8A         1570      TXA -
                  1580      ;
038B- C9 3A      1590      CMP #'
038D- D0 0B      1600      BNE CKEOL
                  1610      ;
                  1620      ;--- CARRY SET ---.
                  1630      ;
038F- 86 5D      1640      STX #IFFLAG
                  1650      ;
0391- A4 5A      1660      LDY #OUTSEG
0393- F0 E3      1670      BEQ SCAN
                  1680      ;
0395- CA         1690      DEX -
0396- 86 5A      1700      STX #OUTSEG
                  1710      ;
                  1720      ;--- CARRY STILL SET ---.
                  1730      ;
0398- 90 A8      1740      NEXTLINJ BCC NEXTLIN
                  1750      ;
039A- AA         1760      CKEOL TAX -
039B- F0 2A      1770      BEQ OUTTEXT
                  1780      ;
039D- E6 5A      1790      INC #OUTSEG
                  1800      ;
039F- A4 5C      1810      LDY #PRVOUT
                  1820      ;
03A1- C9 8B      1830      CKIF CMP #IFTK
03A3- D0 02      1840      BNE CKGO
                  1850      ;
03A5- 85 5D      1860      STA #IFFLAG
                  1870      ;
03A7- C9 CB      1880      CKGO CMP #GOTK
03A9- D0 02      1890      BNE CKTO
                  1900      ;
03AB- A2 89      1910      LDA #GOTOTY
                  1920      ;
03AD- C9 A4      1930      CKTO CMP #TOTK
03AF- D0 0B      1940      BNE CKIFGO
                  1950      ;
03B1- C0 89      1960      CPY #GOTOTK
03B3- F0 C3      1970      BEQ SCAN
                  1980      ;

```

Listing 1 (continued)

```

03B5- C0 A7 1990      CPY #THENTK      ;PRECEDED BY THEN TOKEN ?
03B7- F0 BF 2000      BEQ SCAN          ;YES .. IGNORE INPUT TO TO
2010 ;
03B9- A6 50 2020      CKIF60          LDX #IFFLAG      ;IF TOKEN FOUND ?
03BB- F0 0A 2030      BEQ OUTTEXT      ;NO ... COPY TEXT CHARACER
2040 ;
03BD- C9 89 2050      CKGOTO          CMP #GOTOTK      ;TEXT = GOTO TOKEN ?
03BF- D0 06 2060      BNE OUTTEXT      ;NO ... COPY TEXT CHARACTE
2070 ;
03C1- C0 A7 2080      CPY #THENTK      ;PRECEDED BY THEN TOKEN ?
03C3- F0 B3 2090      BEQ SCAN          ;YES .. IGNORE INPUT GOTO
2100 ;
03C5- A9 A7 2110      LDA #THENTK      ;YES .. REPL. GOTO BY THEN
2120 ;
03C7- A4 59 2130      OUTTEXT          LDY #OUTIND      ;COPY TEXT CHARACTER TO OU
03C9- 91 2A 2140      STA <OUTPTR>.Y    ;SAVE AS PREVIOUS OUTPUT C
03CB- 85 5C 2150      STA #PRVOUT      ;BOOST OUTPUT TEXT INDEX.
03CD- E6 59 2160      INC #OUTIND
2170 ;
03CF- C9 22 2180      CMP #'"          ;A BASIC QUOT COPIED ?
03D1- D0 04 2190      BNE CKEND        ;NO ... CONTINUE
2200 ;
03D3- 45 5B 2210      EOR #QTFLAG      ;SET BASIC QUOT FOUND FLAG
03D5- 85 5B 2220      STA #QTFLAG      ;TO EITHER ON OR OFF.
2230 ;
03D7- A5 5C 2240      CKEND            LDA #PRVOUT      ;END-OF-LINE REACHED ?
03D9- D0 9D 2250      BNE SCAN          ;NO ... CONTINUE SCAN.
2260 ;
2270 ;OUTPUT TEXT LINE CLEANUP
2280 ;
03DB- C0 05 2290      CLEANUP          CPY #5           ;ANY OUTPUT LINE CHARACTER
03DD- 90 11 2300      BCC NEXTIN       ;NO ... DELETE LINE.
2310 ;
2320 ;--- CARRY SET ---.
2330 ;
03DF- A6 5A 2340      LDX #OUTSEG      ;ANY OUTPUT LINE SEGMENT C
03E1- D0 04 2350      BNE NEXTOUT      ;YES .. VALID LINE.
2360 ;
03E3- 08 2370      DELCHR            DEY -             ;DELETE LAST OUTPUT CHARAC
03E4- 8A 2380      TXA -              TXA -
03E5- 91 2A 2390      STA <OUTPTR>.Y    ;
2400 ;
03E7- 98 2410      NEXTOUT            TYA -
03E8- 65 2A 2420      ADC #OUTPTR      ;WITH CARRY SET = <A>+1+OU
03EA- 85 2A 2430      STA #OUTPTR      ;
03EC- 90 02 2440      BCC NEXTIN       ;
03EE- E6 2B 2450      INC #OUTPTR+1    ;
2460 ;
2470 ;GET THE NEXT BASIC INPUT LINE POINTER.
2480 ;
03F0- A5 56 2490      NEXTIN          LDA #NXTLIN
03F2- A4 57 2500      LDY #NXTLIN+1
2510 ;
03F4- 18 2520      CLC
03F5- 90 A1 2530      BCC NEXTLINJ     ;AND CONTINUE SQUEEZING.
2540 ;
2550      .EN

```

Listing 2: Version for BASIC 1.0
Original ROM

```

000 A9 01 85 7C A0 04 84 7D
008 85 A6 84 A7 A0 00 A2 00
010 B1 A6 91 7C 99 A8 00 96
018 AC C8 C0 04 90 F2 84 AA
020 84 AB A0 01 B1 7C D0 16
028 A2 05 A4 7D A5 7C 69 01
030 90 01 C8 94 7C CA 95 7C
038 CA 10 F8 4C 30 C4 A4 AA
040 B1 A6 E6 AA A6 AD D0 45
048 C9 20 F0 F2 C9 8F D0 01
050 8A C9 3A D0 0B 86 AF A4
058 AC F0 E3 CA 86 AC 90 A8
060 AA F0 2A E6 AC A4 AE C9
068 8B D0 02 85 AF C9 C6 D0
070 02 A9 89 C9 A4 D0 08 C0
078 89 F0 C3 C0 A7 F0 BF A6
080 AF F0 0A C9 89 D0 06 C0
088 A7 F0 B3 A9 A7 A4 8B 91
090 7C 85 AE E6 AB C9 22 D0
098 04 45 AD 85 AD A5 AE D0
0A0 9D C0 05 90 11 A6 AC D0
0A8 04 88 8A 91 7C 98 65 7C
0B0 85 7C 90 02 E6 7D A5 A8
0B8 A4 A9 18 90 A1

```

Listing 3: Version for BASIC 3.0
Upgrade ROM

```

000 A9 01 85 2A A0 04 84 2B
008 85 54 84 55 A0 00 A2 00
010 B1 54 91 2A 99 56 00 96
018 5A C8 C0 04 90 F2 84 58
020 84 59 A0 01 B1 2A D0 16
028 A2 05 A4 2B A5 2A 69 01
030 90 01 C8 94 2A CA 95 2A
038 CA 10 F8 4C 42 C4 A4 58
040 B1 54 E6 58 A6 5B D0 45
048 C9 20 F0 F2 C9 8F D0 01
050 8A C9 3A D0 0B 86 5D A4
058 5A F0 E3 CA 86 5A 90 A8
060 AA F0 2A E6 5A A4 5C C9
068 8B D0 02 85 5D C9 C6 D0
070 02 A9 89 C9 A4 D0 08 C0
078 89 F0 C3 C0 A7 F0 BF A6
080 5D F0 0A C9 89 D0 06 C0
088 A7 F0 B3 A9 A7 A4 59 91
090 2A 85 5C E6 59 C9 22 D0
098 04 45 5B 85 5B A5 5C D0
0A0 9D C0 05 90 11 A6 5A D0
0A8 04 88 8A 91 2A 98 65 2A
0B0 85 2A 90 02 E6 2B A5 56
0B8 A4 57 18 90 A1

```

MICRO

Our Current Best-Seller

MICRO
on the Apple

Volume 3 INCLUDES DISKETTE



\$24.95*

More than 40 new programs on
diskette to help you get more from
your Apple:

- Machine-Language Aids
- I/O Enhancements
- Applesoft Aids
- Graphics and Games
- Reference Information

19 choice articles
43 tested programs on diskette
(16 sector DOS 3.3 format)

Volumes 1 & 2 also available at \$24.95*

Together **MICRO on the Apple 1, 2, & 3** provide more than 110 programs on diskette for less than \$1.00 each. No need to type in hundreds of lines of code.

MICRO makes it easy to order:
Send check (payable to MICRO) to:

MICRO INK
P.O. Box 6502
Chelmsford, MA 01824

Call our toll-free number:

1-800-345-8112

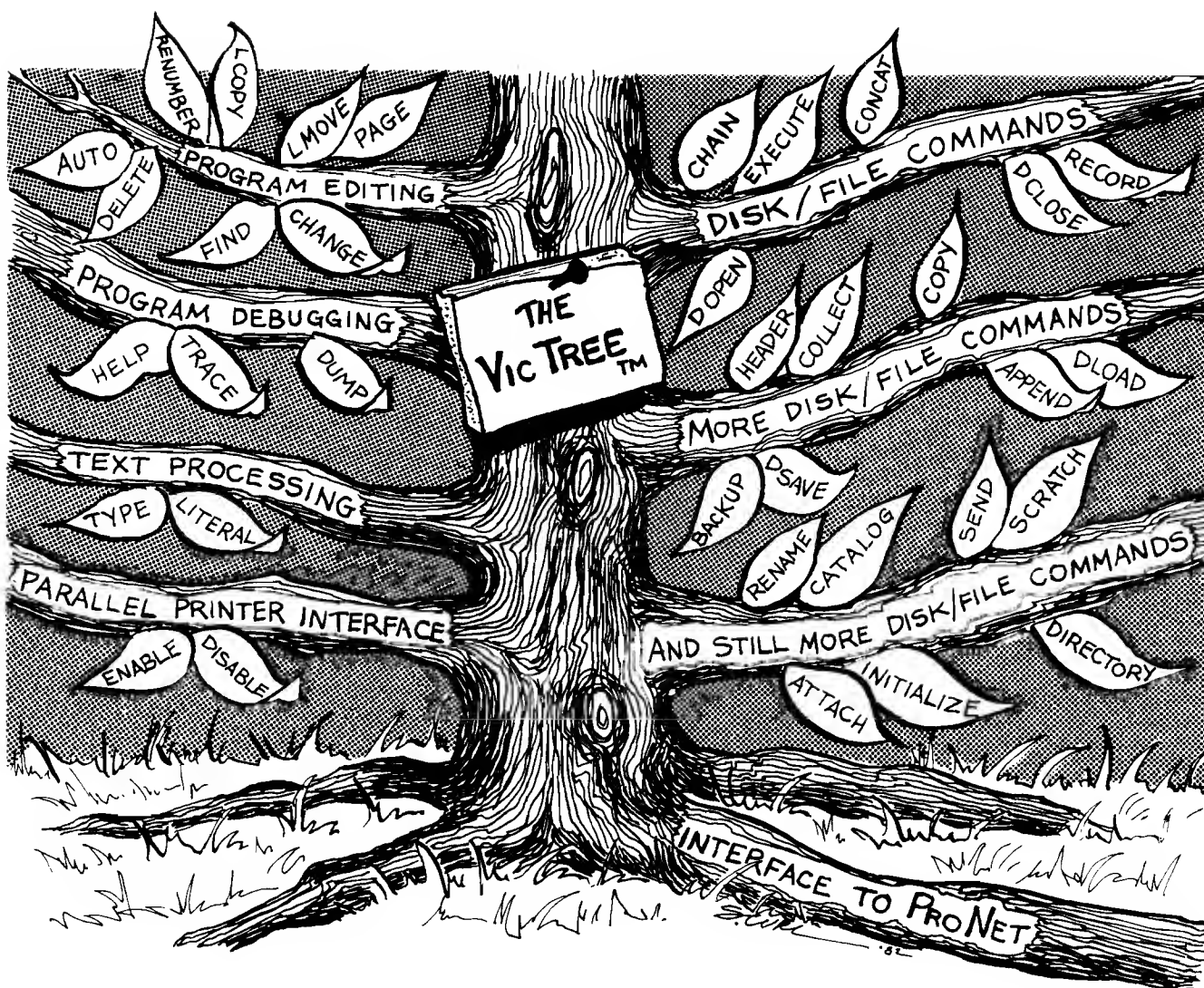
(In PA, 1-800-662-2444)

VISA and MasterCard accepted

Also available at your local computer store.

*Add \$2.00 shipping per book.
MA residents add 5%.

Skyles Electric Works Presents



The VicTree™

...Leaves your new Vic (or CBM 64) with 35 additional commands.

...Branches out to most BASIC 4.0 programs.

...Roots into most printers.

New from Skyles: the VicTree, a coordinated hardware and software package that allows your Vic to branch out in unbelievable directions and makes it easier than ever to do BASIC programming, debugging and to access your disk. And the new VicTree provides routines to interface the Vic to the powerful ProNet local network. 8kb of ROM — 4kb for the BASIC commands, 4kb for disk commands and interfacing to ProNet — plus 4kb of RAM for miscellaneous storage. Perfect not only for the new Vic but also for the Commodore 64. Unbelievably simple to use and to install, the VicTree gives you all the additional BASIC 4.0 commands to allow most BASIC 4.0 programs to work on your new Vic or CBM 64.

Now only \$89.95...or \$99.95 complete with Centronics standard printer cable. (Cable alone \$19.95.) Available now from your local dealer or order through your Visa or MasterCard toll free: (800) 227-9998 (California, Canada, Alaska, Hawaii: (415) 965-1735) or send check or money order directly to:



Skyles Electric Works

231E South Whisman Road
Mountain View, CA 94041
(415) 965-1735

BASIC Line Delete for PET/CBM and VIC

by Thomas Henry

Use this convenient utility during your BASIC program development. It allows you to delete a whole range of lines, rather than just one at a time.

BASIC Line Delete requires:

Upgrade or 4.0 PET/CBM or VIC

"BASIC Line Delete," a command you can add to your Commodore computer's resident BASIC, deletes blocks of BASIC lines instantly. For example, suppose you wish to delete line numbers 1000 through 5000 in a BASIC program. Simply type "<1000-5000" and hit [return] and all those lines will be deleted instantly! This BASIC Line Delete function is easy to use since the syntax is the same as that found for the LIST command. In addition, extensive error checking is employed to avoid disasters.

You can consider BASIC Line Delete as an addition to the computer's BASIC language. It is loaded into the computer at the start of a session and can be invoked at any time, in the immediate mode, to perform its task. Because this 177 byte-long machine-language program sits at the top of memory with memory pointers lowered accordingly, it can peacefully coexist with any BASIC program.

The original program was written on a CBM-8032 with 4.0 ROMs. However, it should be easy to convert to any type of Commodore computer since the ROM routines used are common to all models — only the addresses are different. In addition, it is likely that other Microsoft BASIC machines can use this program with a few changes. When we examine the ROM routines you will note that they are routines that any BASIC interpreter must have.

VIC-20 owners shouldn't feel left out either. Even though the program is in machine language, the VIC-20 can

still use it simply by employing a BASIC loader that POKes the required data into memory. I will present a program to do this later in the article.

Even if you don't want or need a BASIC Line Delete, you may want to look over the program description anyway. Several interesting routines are presented that could be put to other uses. In addition, you may want to see how the program implements error checking and apply it to your own work.

Format of the New Command

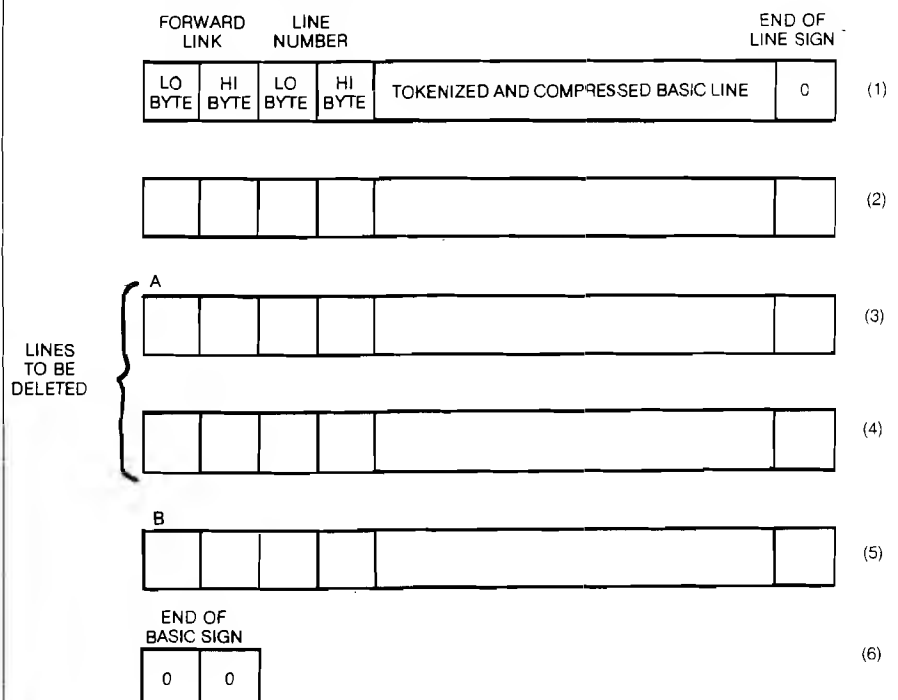
To get a feel for how the program works, let's examine how it should look to the user. The "<" sign indicates the function, although other keys could be used by making one small change in the program. As mentioned before, the format is identical to that used for the LIST command. Let's summarize all proper uses of the BASIC Line Delete:

Proper	Improper
<100-200	<
<100-	<-
<-200	<100
	<--
	etc.

The first statement under proper syntax will delete lines 100 through 200 inclusive. The second one will delete all statements from 100 on. The last one will delete all statements up to line 200 inclusive. And just like the LIST command, there doesn't have to be any line number 100 or 200 for this to work. Suppose the first line number past 90 in your program is actually 122 and the last one before line 210 is 186. Then "<100-200" will still delete all of the lines between this range, meaning that actually lines 122 through 186 are deleted.

The second column shows some of the possible statements with improper

Figure 1 How BASIC is Stored and Principle of DELETE



syntax. If you type any of these, the operation will be aborted and a "?SYNTAX ERROR" message will be returned. It is important to have this feature since a delete function could have potentially catastrophic results if improperly used. So, essentially the statements shown in column one all have proper syntax and will produce meaningful results from the computer, while all other statements will not execute and will produce a syntax error message.

If the range is "backwards" (e.g., <200-100), an error message will again be produced. Finally, I feel so strongly about error checking that I incorporated one more feature. After entering a valid delete command, the computer will respond with "ARE YOU SURE?", giving you one last chance to change your mind! This feature is only available to users with 4.0 operating systems since the "ARE YOU SURE?" routine is part of the normal SCRATCH and HEADER commands.

About the Program

Figure 1 illustrates the principle. As you probably know, a BASIC line is stored in the computer in a specific form. As shown in the illustration, two bytes are devoted to storing the forward link address, which is nothing more than a pointer to the following line in memory. The next two bytes contain the line number. The next area, variable in size, contains the compressed or tokenized BASIC statement. This is polished off with a zero byte to indicate the end of a line. This format is followed throughout memory until the last line is hit. A pair of zeros is included at the end of the last line to indicate the end of the program. (Actually there are three zeros here, if you count the normal end-of-the-line zero). Suppose we wish to delete lines 3 and 4 as indicated in figure 1. What we will do is pick up everything from point B to the end of BASIC and put it back down again at point A. Lines 3 and 4 will be written over in this step. At this point we have just transferred some memory. The link addresses will now be all wrong for the new locations. Fortunately, there is a routine in the ROMs that will rebuild the link addresses for us automatically. After this routine is called the delete has been performed and the BASIC program is all set to go again!

Figure 2 is an assembler listing of the BASIC Line Delete program. As mentioned above, the error checking is the only hard part of the program; the

Figure 2

```

00001 0000
00002 0000
00003 0000
00004 0000
00005 0000
00006 0000
00007 0000
00008 0000
00009 0000
00010 0000
00011 0000
00012 0000
00013 0000
00014 0000
00015 0000
00016 0000
00017 0000
00018 0000
00019 0000
00020 0000
00021 0000
00022 0000
00023 0000
00024 0000
00025 0000
00026 0000
00027 0000
00028 0000
00029 0000
00030 0000
00031 7F52
00032 7F52 A9 4C
00033 7F54 85 79
00034 7F56 A9 63
00035 7F58 85 34
00036 7F5A 85 7A
00037 7F5C A9 7F
00038 7F5E 85 35
00039 7F60 85 7B
00040 7F62 60
00041 7F63
00042 7F63
00043 7F63 C9 3C
00044 7F65 D0 08
00045 7F67 48
00046 7F68 A5 77
00047 7F6A C9 00
00048 7F6C F0 09
00049 7F6E 68
00050 7F6F C9 3A
00051 7F71 90 01
00052 7F73 60
00053 7F74 4C 7D 00
00054 7F77
00055 7F77
00056 7F77 20 70 00
00057 7F7A 90 0D
00058 7F7C F0 1E
00059 7F7E C9 2D
00060 7F80 D0 1E
00061 7F82 20 70 00
00062 7F85 C9 2D
00063 7F87 F0 73
00064 7F89 20 F6 B8
00065 7F8C 20 A3 B5
00066 7F8F A6 5C
00067 7F91 A4 5D
00068 7F93 86 59
00069 7F95 84 5A
00070 7F97 20 76 00
00071 7F9A 90 13
00072 7F9C F0 5E
00073 7F9E C9 2D
00074 7FA0 D0 5A
00075 7FA2 20 70 00
00076 7FA5 D0 08
00077 7FA7 A2 FF
00078 7FA9 86 11
00079 7FAB 86 12
00080 7FAD D0 03
00081 7FAF 20 F6 B8
00082 7FB2 20 A3 B5
00083 7FB5 90 0C
00084 7FB7 A0 00
00085 7FB9 B1 5C
00086 7FBB AA
00087 7FBC C8
00088 7FBD B1 5C
00089 7FBF 86 5C
00090 7FC1 85 5D
00091 7FC3 38
00092 7FC4 A5 5C
00093 7FC6 E5 59

*****
;*
;* BASIC LINE DELETE UTILITY
;*
;* ASSEMBLER CODE FOR CBM-8032
;* THOMAS HENRY
;*
*****
;
;
VALUE = $11 ;INTEGER VALUE.
VARBLE = $2A ;POINTER TO VARIABLES.
MEMTOP = $34 ;TOP OF MEMORY POINTER.
SAVE = $59 ;SAVE START ADDRESS.
ADDRES = $5C ;ADDRESS OF FOUND LINE #.
CHRGOT = $70 ;BASIC CHRGET ROUTINE.
CHRGOT = $76 ;BASIC CHRGET ROUTINE.
POINTR = $77 ;CHRGET POINTER.
WEDGE = $79 ;WEDGE GOES HERE.
RETURN = $7D ;RETURN TO CHRGET ROUTINE.
FIXUP = $B4AD ;ADJUST POINTERS.
CHAIN = $B4B6 ;REBUILD LINE CHAINING.
SEARCH = $B5A3 ;SEARCH FOR BASIC LINE.
INTEGR = $B8F6 ;FETCH INTEGER INPUT.
ERROR = $BF00 ;SYNTAX ERROR ROUTINE.
QUERY = $DB9E ;'ARE YOU SURE?'
CHROUT = $E202 ;PRINT CHARACTER TO SCREEN.
;
;
* = $7F52
;
SETUP LDA #$4C ;OP-CODE FOR 'JMP'.
STA WEDGE
LDA #CENTRY ;LOW BYTE OF ENTRY.
STA MEMTOP ;LOWER MEMORY TO PROTECT.
STA WEDGE+1
LDA #>ENTRY ;HIGH BYTE OF ENTRY.
STA MEMTOP+1 ;LOWER MEMORY TO PROTECT.
STA WEDGE+2
RTS ;INITIALIZATION COMPLETE.
;
;
ENTRY CMP #< ;LOOK FOR DELETE SYMBOL.
BNE COMMON ;SORRY, NOT HERE.
PHA ;YES, IT'S HERE. SAVE.
LDA POINTR
CMP #$00 ;CHECK FOR IMMEDIATE MODE.
BEQ DELETE ;DO DELETE IF IMMEDIATE.
PLA ;DON'T DO IN PROGRAM MODE.
COMMON CMP #$3A ;COMPLETE CHRGET ROUTINE.
BCC FINISH
RTS
FINISH JMP RETURN
;
;
DELETE JSR CHRGET ;FETCH FIRST CHARACTER.
BCC FIRST ;IT'S A NUMBER.
BEQ MIDDLE ;NULL INPUT IS ERROR.
CMP #'- ;IS IT A MINUS SIGN?
BNE BYPASS ;NO, ERROR!
JSR CHRGET ;FETCH NEXT CHARACTER.
CMP #'- ;IS IT ANOTHER MINUS SIGN?
BEQ BAD ;IF IT IS, THEN ERROR.
JSR INTEGR ;ACCEPT INTEGER INPUT.
JSR SEARCH ;FIND THE LINE NUMBER,
LDX ADDRESS ;AND SAVE ITS ADDRESS.
LDY ADDRESS+1
STX SAVE
STY SAVE+1
JSR CHRGOT ;LOOK AGAIN AT CHAR.
BCC LAST ;GO GET LAST LINE NUMBER.
MIDDLE BEQ BAD
CMP #'- ;IS IT A MINUS SIGN?
BNE BYPASS ;NO, ERROR!
JSR CHRGET ;YES, FETCH NEXT CHAR.
BNE LAST ;IF PRESENT, GO ON.
LDX #$FF ;OTHERWISE DEFAULT TO
STX VALUE ;LINE NUMBER $FFFF.
BNE DEFALT ;BRANCH ALWAYS.
LAST JSR INTEGR ;GET LAST LINE #.
DEFALT JSR SEARCH ;FIND ADDRESS OF LINE #.
BCC CHECK ;BRANCH, LINE NOT FOUND.
LDY #$00
LDA (ADDRESS),Y ;GET FORWARD LINK TO
TAX ;POINT TO NEXT LINE IN
INY ;MEMORY.
LDA (ADDRESS),Y
STX ADDRESS
STA ADDRESS+1
CHECK SEC ;CHECK TO SEE THAT THE
LDA ADDRESS ;START NUMBER IS LOWER
SBC SAVE ;THAN THE STOP NUMBER.

```

Figure 2 (continued)

00094	7FCB	A5 5D		LDA ADDRESS+1	
00095	7FCA	E5 5A		SBC SAVE+1	
00096	7FCC	90 2E		BCC BAD	
00097	7FCE	20 9E DB		JSR QUERY	;IT'S NOT, SO ERROR.
00098	7FD1	B0 21		BCS DONE	;IT IS. LAST CHANCE
00099	7FD3	A0 00	MOVE	LDY #*00	;TO CHANGE YOUR MIND.
00100	7FD5	B1 5C		LDA (ADDRESS),Y	;SHIFT BYTES BACK,
00101	7FD7	91 59		STA (SAVE),Y	;ONE BY ONE.
00102	7FD9	E6 59		INC SAVE	;INCREMENT START ADDRESS.
00103	7FDB	D0 02		BNE NOCAR1	
00104	7FDD	E6 5A		INC SAVE+1	
00105	7FDF	E6 5C	NOCAR1	INC ADDRESS	;INCREMENT END ADDRESS.
00106	7FE1	D0 02		BNE NOCAR2	
00107	7FE3	E6 5D		INC ADDRESS+1	
00108	7FE5	A5 5C	NOCAR2	LDA ADDRESS	;IS END ADDRESS TOUCHING
00109	7FE7	C5 2A		CMP VARBLE	;THE START OF VARIABLES YET?
00110	7FE9	D0 E8		BNE MOVE	;IF IT ISN'T, DO MORE.
00111	7FEB	A5 5D		LDA ADDRESS+1	
00112	7FED	C5 2B		CMP VARBLE+1	
00113	7FEF	D0 E2		BNE MOVE	
00114	7FF1	20 B6 B4		JSR CHAIN	;REBUILD CHAINING OF LINES.
00115	7FF4	A9 0D	DONE	LDA #*0D	;PRINT CARRIAGE RETURN.
00116	7FF6	20 02 E2		JSR CHROUT	
00117	7FF9	4C AD B4		JMP FIXUP	;CLEAN UP POINTERS, ETC.
00118	7FFC	4C 00 BF	BAD	JMP ERROR	
00119	7FFF			.END	

delete part is quite easy. I will let you examine the assembler listing, but as an aid to understanding, let me describe the key ROM routines used in it. You may want to jot these down in your notebook for future reference, since I'm sure these routines have many more valuable uses.

The routine at \$B8F6 will get an integer from the screen. The CHRGET

routine (at \$70) is called first and this causes locations \$77 and \$78 to point to the start of the integer (which is in ASCII). After a JSR \$B8F6, the ASCII representation is converted to a binary form and the result is deposited in locations \$11 and \$12 (low byte and high byte, respectively). If \$77 and \$78 point to the "-" sign (as in the command "<-200"), the subroutine will return

with zeros in \$11 and \$12. You can consider this as a default lower line number.

Given a line number, routine \$B5A3 will find where in memory that BASIC line sits. Simply put the desired line number in locations \$11 and \$12 and call routine \$B5A3. The routine will return with the address of the first byte of the desired line in locations \$5C and \$5D. You will note that the routine described in the preceding paragraph ends with the desired data in locations \$11 and \$12, whereas this routine begins with data in these locations. This means that we can chain the two routines without saving any intermediate results!

An interesting feature of this line-finding routine is its ability to adapt to non-existent line numbers. For example, suppose you tell it to find line 100 but no such number exists in your program. However, your program does contain a statement with line number 110. When you call the routine it will look for number 100 and won't find it. But it will continue to look for the first line number beyond 100 (in this case 110) and return with its address in-

The
MIDNITE
SOFTWARE GAZETTE

The
PAPER

Five years of service to the PET community.



The Independent U.S. magazine for users of Commodore brand computers.

EDITORS: Jim and Ellen Strasma
Sample issue free on request, from:

635 MAPLE □ MT. ZION, IL 62549 USA

COMPU SENSE

CARDC Products For the Vic-20®

VIC-20® is a registered trademark of Commodore.

CARDBOARD 6 Expand your Vic-20® to accept 6 game cartridges or an additional 35K of RAM memory. Until December 15, 1982, a SPECIAL CHRISTMAS PRICE \$79.95, thereafter \$99.95

CARDBOARD 3 An economy memory expander to 35KAM or 3 game cartridges \$29.95

CARDPRINT Printer interface—Vic to Centronics type parallel input printer. This unit uses the regular Vic printer out put port and uses normal Vic print commands \$79.95

CARDRITER A light pen with 6 software programs including a full screen drawing tablet, a light pen instruction program, and some games \$29.95

Cardette universal cassette interface for the Vic. Allows any standard cassette player/recorder to be used in place of the Vic Dattasette, \$29.95

We carry a full line of Vic-20® hardware and software. We have a full line of other software including the exciting NEW SOFT\$ENSE LINE of Small Business or personal business programs.

SEND FOR OUR CATALOG

To Order:

P.O. Box 18765
Wichita, KS 67218
(316) 684-9660

Personal Checks Accepted (Allow 3 Weeks), or C.O.D.
Handling Charges \$1.50

Prices Subject to Change without notice



stead. You can see that this is exactly what the BASIC Line Delete program needs! One other feature is that if the exact line number specified was found, then the carry flag is set. Otherwise, as in our example here, the carry flag will be cleared.

In the program, if no last line number is specified, a default number of \$FFFF (65535 decimal) is specified. Notice what happens when this number is acted on by subroutine \$B5A3. Suppose the actual last number in your BASIC program is 1000 and you enter the command "<250-". The default number \$FFFF is loaded into \$11 and \$12 and routine \$B5A3 is called. The routine will start with 65535 and will whittle away at the numbers until it eventually hits your actual last number (1000 in this case). Once again, this is exactly what the BASIC Line Delete requires.

The routine at \$DB9E will query "ARE YOU SURE?" and wait for a reply. If the answer is "Y" or "YES" the carry flag will be cleared. Any other response will set the carry flag. Note

Figure 3

```

100 REM *****
110 REM *
120 REM *          BASIC LINE DELETE:
130 REM *          VIC-20 VERSION
140 REM *
150 REM *          THOMAS HENRY
160 REM *
170 REM *          TRANSONIC LABORATORIES
180 REM *          249 NORTON STREET
190 REM *          MANKATO, MN 56001
200 REM *
210 REM *****
220 REM
230 REM
240 PRINT "WAIT A MOMENT..."
250 X=PEEK(55)+256*PEEK(56)-163
260 FORA=X TO X+162
270 READD:POKEA,D:NEXT
280 Y=X+17:H%=Y/256:L=Y-256*H%
290 POKEX+5,L:POKEX+11,H%
300 SYS(X):NEW
310 DATA169,76,133,124,169,110,133,55,133,125,169,29,133,56,133,126
320 DATA96,201,60,208,8,72,165,122,201,0,240,9,104,201,58,144
330 DATA1,96,76,128,0,32,115,0,144,13,240,116,201,45,208,112
340 DATA32,115,0,201,45,240,105,32,107,201,32,19,198,166,95,164
350 DATA96,134,92,132,93,32,121,0,144,19,240,84,201,45,208,80
360 DATA32,115,0,208,8,162,255,134,20,134,21,208,3,32,107,201
370 DATA32,19,198,144,12,160,0,177,95,170,200,177,95,134,95,133
380 DATA96,56,165,95,229,92,165,96,229,93,144,36,160,0,177,95
390 DATA145,92,230,92,208,2,230,93,230,95,208,2,230,96,165,95
400 DATA197,45,208,232,165,96,197,46,208,226,32,51,197,76,42,197
410 DATA76,8,207

```

that due to a quirk in this routine, you should print a carriage return to the screen following it. This will move the cursor to the proper position on the next line. To print a carriage return, do the following:

```

LDA #$0D
JSR $E202

```

To rebuild the forward link chain- ing, simply call subroutine \$B4B6. No set-up is needed to enter this routine.

The BASIC Line Delete program ends with two alternate ways to get back into BASIC. If JMP \$B4AD is used, then a graceful return will be made to BASIC, indicating that all went well. However, if a return is made via JMP \$BFO0, the statement "SYNTAX ERROR" will be printed indicating that the attempted operation was aborted.

To round out your survey of this program note that locations \$59 and \$5A hold the address of the start line number (where the later memory will be moved to; "A" in figure 1). \$5C and \$5D hold the address of the end line ("B" in figure 1). \$2A and \$2B are pointers to the end of BASIC.

How to Load and Use the Program

If you have a computer other than 4.0, you will have to make the required translations to your machine. If you have memory maps handy this shouldn't take too long. I was able to make a VIC-20 version in about fifteen

minutes simply by comparing memory maps. Just enter the resident machine-language monitor and list out the required lines with the command:

```
.M 7F52,7FFF
```

Now type over what the computer shows, using the byte values generated in the assembly in figure 2 as a guide. When you are done, save the program with the command:

```
.S "DELETE - 32594",08,7F52,7FFF
```

If you are saving to tape replace the "08" with an "01". The number in the title is the SYS number.

Suppose you are using the program at the start of a session (from a cold start). First LOAD the program in the normal way (just like a BASIC program). There is no need to load it from the monitor; the CBM-8032 knows where to put it. Next type NEW and hit return. This step is important since it resets some pointers previously disarrayed by the LOAD command. Now type SYS32594 and hit return. The BASIC Line Delete is now activated. The top of memory pointers are automatically lowered to protect it. You are now free to call up the function whenever desired.

This program is very relocatable. If you decide to put it somewhere else in memory only locations \$7F57 and \$7F5D need be changed. These two bytes form the address of the CHRGET

VIC-20

VIC-20 INTERFACING BLUE BOOK

Did you know that your VIC can be used to control a 99c toy motor so effectively that it runs like a precision machine? Or that you can build an accurate digital thermometer using the VIC and four other parts costing less than \$5?

These and other 18 interfacing projects selected for usefulness, ease of construction and low cost are detailed in the VIC-20 Interfacing Blue Book, a veritable gold mine of practical information on how to build a variety of interfaces for your computer.

Projects include: Connecting VIC to your stereo; Pickproof digital lock; Capacitance meter; Liquid level sensor; Telephone dialer; Voice output; 8K/16K RAM/ROM expansion; 128K RAM expansion; 8 bit precision D/A; 8-bit A/D converter; MX-80 printer interface and more.

Written by a college professor in a friendly and informative style, the Blue Book gives you theory of operation, schematics, program listings, parts list, construction hints and sources of materials for each one of the 20 projects.

If you want to get the most out of your VIC this book is a must. Even if you don't plan to build any of the projects, the Blue Book is a valuable source of information on what can be done with the VIC. Cost is \$14.95 (less than 75¢ per project!).

WORD WHIZ

Here is a no-frills word processor that does the job and is so small it leaves plenty of memory for your text. Yet it offers full screen editing and easy save of work in progress on cassette, by taking advantage of VIC's built-in text manipulation capabilities. WORD WHIZ prints out on the 1515 printer and is a bargain at \$9.95.

WORD WHIZ/80

For classy looking output, this version of WORD WHIZ will drive an EPSON MX-80 (See Interfacing info in Blue Book above). Get letter quality printing for only \$14.95.

Above prices include postage in the U.S. CA res. add 6% tax. Foreign add \$2.

microsignal

900 Embarcadero Del Mar, Unit A
Goleta, CA 93117

VIC-20

Add-on, starting at \$7F63 in this case. Everything else remains the same. This is due to extensive use of relative addressing; there are no internal JSR or JMP commands to be altered. Simply transfer the program, change the two bytes mentioned, and run it using the new SYS address!

VIC-20 owners need a different way to get the program into memory since the VIC has no resident machine-language monitor. Figure 3 shows a loader program that will enter an equivalent BASIC Line Delete into memory. Note that this loader is completely automatic since it not only loads the program but also instantly adjusts to VIC-20s with any amount of add-on memory. In addition, the program automatically does a SYS to the right address. All the user has to do is LOAD the program and RUN it!

Now you have a new command for your Commodore computer. You don't really have to understand how it works to use it, but I recommend you look over the assembly listing again. As mentioned before, the ROM routines

called are quite powerful and probably have many other uses. In addition, the program itself could serve as an example of how to incorporate worst-case error checking into your own routines.

Acknowledgements

I owe a big debt of gratitude to Dick Immers of the Central Illinois PET User's Group for explaining some of the quirks of the CBM-8032 machine-language tape-save routine. Thanks also go to Dr. Kenneth Good, Mankato State University, for putting early versions of this program to the acid test. He found several conditions that could have caused users real troubles were they not flagged with "SYNTAX ERROR" statements.

Thomas Henry is a professional writer in the areas of electronic music, circuit design, and Commodore computers. He may be contacted at Transonic Laboratories, 249 Norton Street, Mankato, MN 56001.

MICRO

70 INCOME TAX PROGRAMS

(For Filing by April 15, 1983)

For APPLE II/II* (DOS 3.3, 16-Sector)

FEATURES:—

1. Menu Driven.
2. 70+ Tax Programs.
3. Basic; Unlocked; Listable.
4. Name/SS No./FS carried over.
5. Inputs can be checked.
6. Inputs can be changed.
7. I.R.S. approved REVPROC format.
8. Prints entire Form/Schedule.
9. Calculates Taxes, etc.
10. In 3.3 DOS, 16-Sector.
11. Fast calculations.
12. Use GREENBAR in triplicate — don't change paper all season!
13. Our 4th Year in Tax Programs.
14. We back up our Programs!

Helpful programs to calculate and print the many Tax Forms and Schedules. Ideal for the Tax Preparer, C.P.A. and Individuals. For just \$24.75 per disk, post-paid (in 3.3 DOS; 16-Sector disks).

Programs are designed for easy-use, with checkpoints to correct parts as needed. Results on screen for checking before printing.

In all, there are more than 70 individual Tax Programs. These include Form 1040, 1040A, 1040EZ, 1120, 1120S, 1041 and 1065. Also Schedules A, B, C, D, E, F, G, R, RP and SE. And, Forms 1116, 2106, 2119, 2210, 2440, 3468, 3903, 4255, 4562, 4797, 4835, 4972, 5695, 6251 and 6252.

And, we have a disk we call "THE TAX PREPARER'S HELPER" which has programs for INCOME STATEMENTS, RENTAL STATEMENTS, SUPPORTING STATEMENTS, IRA, ACRS, 1040/ES, ADD W-2's and PRINT W-2's.

TRY ONE DISK AND SEE FOR YOURSELF. ONLY \$24.75 POSTPAID.

First disk is AP#1, and includes Form 1040 and Schedules A, B, C, D and G. \$24.75 POSTPAID.

Write:—

GOOTH TAX PROGRAMS

931 So. Bemiston • St. Louis, Mo. 63105



A harvest of savings from



Apple Tree Electronics

SOFTWARE

APPLE • ATARI • TRS80 • IBM

A full line of software for business, games and education **up to 35% off!**

MUSE	KIS
VISICORP	STONEWARE
ON LINE	SYNERGISTIC
EDU-WARE	HAYDEN
HOWARD	AND MANY MORE

HARDWARE

AMDEK • HAYES • MICROSOFT

FRANKLIN COMPUTER SYSTEM

ACE 1000 • \$1,795.00

DISKS

Maxell Box of 10, 5¼", SS-DD \$35.00
Verbatim Box of 10, 5¼", SS-DD \$29.00

MONITORS

LE MONITORS	List	Our Price
9" Green	\$189.00	\$159.00
12" Green	\$199.00	\$169.00
ZENITH		
12" Green	\$179.00	\$129.00

Plus a full line of AMDEK Monitors

PRINTERS

PAPER TIGER	List	Our Price
460G	\$1,094.00	\$950.00
560G	\$1,394.00	\$1,250.00
EPSON		
MX 70	\$449.00	\$395.00
MX 80FT	\$745.00	\$595.00
MX 100FT	\$945.00	\$795.00

CALL FOR THIS MONTH'S SPECIAL!

1-800-835-2246 EXT. 211

OR

702-459-4114



5130 East Charleston Blvd.
Suite 5M1
Las Vegas, Nevada 89122



Phone orders welcome. Mail orders may send charge card number (include expiration date), cashiers check, money order or personal check (allow ten business days for personal or company checks to clear). Add \$3.00 for shipping, handling and insurance. Nevada residents add 5.75% sales tax. Please include phone number. All equipment is in factory cartons with manufacturers warranty. Equipment subject to price change and availability. Call or write for price list.

SOUP: A CBM Machine-Language Compare Program

by Henry Troup and Jim Strasma

SOUP is an efficient compare program for machine-language program files on Commodore disk. It uses BASIC 4.0 disk commands, but is otherwise compatible with other Microsoft BASICs.

SOUP

requires:

PET/CBM
disk drives
printer (optional)

This program, originally adapted by Henry Troup from a similar mini-computer utility, compares two versions of a machine-language program on disk and prints out any lines that differ between the two versions. All you need to use SOUP are disk copies of the two machine-language programs to be compared. The only other restriction is that they must begin loading at the same address.

To use the program, place the disk or disks with the files to compare in your disk unit. Also prepare your printer, if you are using one. At start up, you will be asked the name and drive number of the two files. This is the only time in the program that disk status is checked. If an error is found here, repair the cause and re-enter the file name and drive number.

From here on, operation is automatic. As differences are discovered they are listed either to the screen or printer. You may wish to make some changes in the formatting used here. Lines 700 and 710 set the maximum fields per line for screen and printer respectively. If your screen has over 40 columns, or your printer over 80, you may increase the value given to variable mf. Likewise, if your printer is not device #4,

change lines 690 and 710 to allow the device number you need. If your paper is not the 11-inch variety common in the U.S., change line 350 to adjust the lines printed per page to your needs.

To better explain its workings, the program as printed here is heavily commented and uses fewer multiple statement lines than it could. Feel free to omit remark statements and lines containing only a colon; none is referenced by other lines. You may also be able to combine some lines. For example, the subroutine beginning in line 460 could be reduced to four lines. Likewise, the spaces that are not within quotation marks may safely be left out. However, you may find it better to leave the program as listed here and compile it.

In the interest of speeding up the program, often-used constants are replaced by variables, seldom-used lines are moved to the end of the listing, and disk status is left unchecked once the needed files are successfully opened. If you notice that the program seems to have halted with the disk error light on, hit the [stop] key, and check the disk status in immediate mode:

?ds\$

Most likely the error will be fatal, and you will have to start over again after correcting the problem.

The program uses only a few special characters. In lines 670, 730, 740, 780, and 790 notice the three equal signs in a row (= = =). These represent three [cursor left] characters. These characters place the flashing input cursor over a likely default answer. They also protect the user from accidentally falling out of the program. Even so, you may omit them.

To use this program with other computers or disk drives, you will need only to substitute your disk commands for Commodore's. The most difficult task for other disk operating systems is likely to be reading in the program files one character at a time. The other essential task is to detect the end of file when it is reached. If you know how to do these tasks on your machine, you can probably make SOUP work for you.

Henry Troup and Jim Strasma may be contacted at 1280 Richland Ave., Lincoln, IL 62656.

Listing 1

```
100 REM SOUP -- AS OF 7 SEPT 82
110 GOSUB 630:REM PUT MOST-USED LINES AT START FOR SPEED
120 REM MAIN ROUTINE
130 NM$="SOUP: FILE A="+CF$+" & FILE B="+PF$:REM TITLE
140 PRINT#4,NM$:REM START NEW PAGE
150 GET#1,A$:REM READ A CHARACTER FROM FILE A
160 S1=ST:REM REMEMBER I/O STATUS OF A
170 IF A$=NL$ THEN A$=ZE$:REM TRAP NULL DATA BUG
180 GET#1,B$:REM READ A CHARACTER FROM FILE B
190 S2=ST:REM REMEMBER I/O STATUS OF B
200 IF B$=NL$ THEN B$=ZE$:REM FIX NULL DATA BUG
210 IF A$=B$ GOTO 420:REM ONLY REPORT DIFFERENCES
220 A=ASC(A$):B=ASC(B$):REM CONVERT TO DECIMAL CODE
230 N=AD:GOSUB 490:REM CONVERT ADDRESS TO HEXADECIMAL
240 PRINT#4,"@HX$ ",A=";:REM PRINT MISMATCH
250 N=A:GOSUB 490:REM CONVERT A'S VALUE TO HEX
260 PRINT#4,HX$ "B=";:REM & PRINT IT
270 N=B:GOSUB 490:REM THEN CONVERT B'S
```

```

280 PRINT#4,HX$;:REM & PRINT 4
290 FC=FC+1:REM PRINT 4 MISMATCHES PER LINE
300 REM TAB IF HAVE ROOM FOR ANOTHER ON LINE
310 IF FC<MF THEN PRINT#4," ";:GOTO 420
320 FC=0:REM ELSE RESET FIELD COUNTER
330 PRINT#4:REM & FINISH LINE
340 LC=LC+1:REM INCREMENT LINE COUNTER
350 IF LC<59 THEN 420:REM 58 MISMATCH LINES PER PAGE
360 LC=0:REM RESET LINE COUNTER
370 FOR I=1 TO 6:REM SKIP LAST 6 LINES
380 : PRINT#4
390 NEXT
400 PRINT#4,NM$:REM TITLE NEXT PAGE
410 REM END ON STATUS CHANGE, (END OF FILE)
420 IF S1 OR S2 THEN DCLOSE:PRINT#4:CLOSE 4:END
430 AD=AD+1:REM ELSE INCREMENT ADDRESS COUNTER
440 GOTO 150:REM & CONTINUE
450 :
460 REM DECIMAL TO HEX CONVERTER SUBROUTINE
470 REM ENTER WITH NUMBER IN N
480 REM RETURNS HEX EQUIVALENT IN HX$
490 IF N=0 THEN HX$="00":GOTO 600:REM HANDLE EXCEPTION
500 HX$="":REM INITIALIZE OUTPUT VARIABLE
510 D=-LOG(N)/LOG(16)
520 D%=D-(D<>INT(D))
530 FOR I=D% TO 0:REM LOOP FOR DIGITS
540 : P=16^(-I)
550 : Q%=N/P
560 : HX$=HX$+CHR$(Q%+48-7*(Q%>9))
570 : N=N-Q%*P
580 NEXT
590 IF LEN(HX$)=1 THEN HX$="0"+HX$:REM FORMAT 1 CHARACTER
600 HX$="$"+HX$
610 RETURN
620 REM SETUP SUBROUTINE
630 PRINT"SOUP BY HENRY TROUP & JIM STRASMA
640 PRINT"COMPARES MACHINE-LANGUAGE PROGRAMS
650 REM PRESET VARIABLES TO GAIN SPEED
660 NL$="":ZE$=CHR$(0)
670 INPUT"OUTPUT DEVICE: 3=SCREEN, 4=PRINTER 3===";OTS$
680 DV=VAL(OTS$):REM CONVERT TO NUMBER
690 IF DV<3 OR DV>4 GOTO 670:REM VALIDATE
700 MF=2:REM 2 FIELDS PER LINE ON SCREEN
710 IF DV<>3 THEN MF=4:REM 4 FOR PRINTER
720 CLOSE 4:OPEN 4,DV:REM HELLO DEVICE
730 INPUT"FILE A'S NAME +===";CF$
740 INPUT"ON DRIVE 0===";R1
750 IF R1<>0 AND R1<>1 THEN 740:REM VALIDATE
760 DOPEN#1,(CF$),D(R1):REM HELLO FILE A
770 IF DS THEN PRINT DS$:GOTO 730:REM ON ERROR
780 INPUT"FILE B'S NAME +===";PF$
790 INPUT"ON DRIVE 0===";R2
800 IF R2<>0 AND R2<>1 THEN 790:REM VALIDATE
810 DOPEN#2,(PF$),D(R2):REM HELLO FILE B
820 IF DS THEN PRINT DS$:GOTO 780:REM ON ERROR
830 GET#1,A1$:GET#1,A2$:REM READ A'S LOAD ADDRESS
840 GET#2,B1$:GET#2,B2$:REM & B'S
850 REM TRAP ZERO DATA BUG
860 IF A1$=NL$ THEN A1$=ZE$
870 IF A2$=NL$ THEN A2$=ZE$
880 IF B1$=NL$ THEN B1$=ZE$
890 IF B2$=NL$ THEN B2$=ZE$
900 REM CALCULATE LOAD ADDRESSES
910 AD=ASC(A1$)+ASC(A2$)*256
920 A2=ASC(B1$)+ASC(B2$)*256
930 IF AD=A2 THEN RETURN:REM IF MATCH, BEGIN
940 PRINT"START ADDRESSES DON'T MATCH
950 DCLOSE:REM ELSE CLOSE DISK FILES
960 END:REM & ABORT

```

```
SOUP: FILE A=SOUP      & FILE B=SOUP 7SE82
@S401,A=$1B+B=$04      @S402,A=$64+B=$00      @S403,A=$8F+B=$20      @S405,A=$20+B=$43
@S406,A=$45+8=$2D      @S407,A=$28+B=$51      @S408,A=$44+B=$2C      @S409,A=$4E+B=$2C
@S40A,A=$44+B=$2C      @S40B,A=$50+B=$2C      @S40C,A=$41+B=$32      @S40D,A=$29+B=$50
@S40E,A=$43+B=$04      @S40F,A=$6E+B=$20      @S410,A=$8F+B=$32      @S411,A=$50+B=$52
@S412,A=$49+B=$4F      @S413,A=$52+B=$20      @S414,A=$4C+B=$49      @S415,A=$4E+B=$45
@S416,A=$20+B=$4E      @S418,A=$44+B=$45      @S419,A=$44+B=$20      @S41A,A=$42+B=$59
@S41B,A=$20+B=$44      @S41C,A=$54+B=$4C      @S41D,A=$20+B=$43      @S41E,A=$4F+B=$4D
@S41F,A=$50+B=$49      @S420,A=$4C+B=$45      @S421,A=$52+B=$30      @S422,A=$76+B=$04
```

[illegible]

MICROTM

PET Vet

By Loren Wright

Graphics on the Commodore 64

The Commodore 64 offers a lot of computing power in its small package. There are 64K of RAM, CP/M capability, and sophisticated sound features. But the most outstanding feature is the graphics. To sum it up, the 64 offers considerably more graphics capabilities than the Apple in this area and rivals the Atari 800, at a price that beats them both.

What, exactly, does the 64 do in the way of graphics? I've been studying a preliminary draft of the *Commodore 64 Programmer's Reference Guide* and have begun to learn about all the graphics on my own 64.

The 64 has the following modes, some of which can be mixed on the same screen:

1. Standard character mode
 - a. ROM characters
 - b. Programmable RAM characters
2. Multicolor character mode (both ROM and RAM)
3. Extended background color mode (both ROM and RAM)
4. Standard bit-map mode (320 × 200 resolution)
5. Multicolor bit-map mode (160 × 200 resolution)
6. Sprites (both standard and multicolor modes)

Various blocks of memory and control registers are involved in pulling off all these different modes. Screen memory consists of 1000 bytes, normally located at \$400, and these usually determine what characters will appear on the screen. There is a character ROM, which contains two complete character sets, as on the PET and VIC. Pointers may be altered so that custom characters can be set up in RAM. Color memory, which can't be moved, is

1000 4-bit locations at \$D800, each corresponding to a location in screen memory. Four bits is enough to code for sixteen different colors.

The VIC II uses the different bits of two control registers to select nearly all of the graphics modes. Other registers are used to control positions and colors of sprites, to read light pens, and to select background colors. This month's data sheet (p. 109) lists the control registers for the 64. I will refer to them here only by name.

Character Modes

The 64's characters are normally read from the character ROM and the color is determined by the contents of the corresponding location in color memory. The pointer to the character ROM can be altered to point to RAM, where you can design custom characters. There's plenty of memory to play with, so this is a lot more practical than on an unexpanded VIC!





Multicolor character mode has a lot of possibilities. Standard characters consist of eight rows of eight pixels, while multicolor characters consist of eight rows of four double-width pixels. (A pixel is the smallest dot of light on

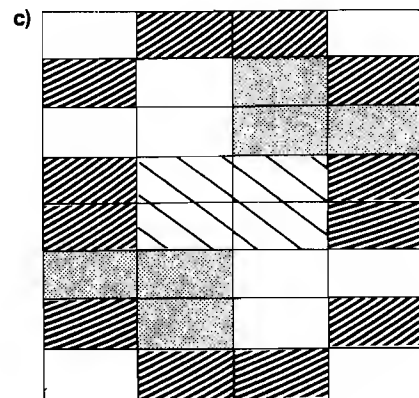
the TV screen in the current graphics mode.) The bits of each byte in character memory are considered in pairs rather than individually. Each of the four possible bit combinations for a bit pair determines where to get the color for the double-wide pixel on the screen. Combinations 00, 01, and 10 get the color from background registers 0, 1, and 2, respectively, and 11 gets the color from the appropriate location in color memory. Since any background color can be changed with a single POKE, parts of all the characters on the screen can be changed at once! This mode is probably best used with custom characters, since this way of interpreting the character data would make most standard characters nearly unrecognizable. The VIC uses a similar scheme in its multicolor mode.

Extended background mode allows the background for each screen location to be any of four different colors. The sacrifice is that only the first 64 characters in character memory can be used. Bits 6 and 7, which would normally select the other 192 characters, determine the background color instead. The background color is read from background color register 0, 1, 2, or 3.

Figure 1. Multicolor Character Mode a) Bits in character memory are considered in pairs. b) Each bit combination indicates a different source for the color. c) The final character displayed with double-width pixels.

a)	0	0	0	1	0	1	0	0
	0	1	0	0	1	0	0	1
	0	0	0	0	1	0	1	0
	0	1	1	1	1	1	0	1
	0	1	1	1	1	1	0	1
	1	0	1	0	0	0	0	0
	0	1	1	0	0	0	0	1
	0	0	0	1	0	1	0	0

b)	00	Background 0	
	01	Background 1	
	10	Background 2	
	11	4-bit color location	



Bit-mapped Modes

Standard bit-map (or high-resolution) mode allows control of each individual pixel on the screen, with a resolution of 320 by 200. 8K of RAM, normally taken from the top of BASIC RAM, is used for high-resolution graphics. The bytes are arranged in the same way the pixels of characters are coded. That is, the first byte in hi-res memory codes for the first eight pixels in the first row of pixels on the screen, and the second codes for the first eight pixels in the second row. The ninth byte codes for the ninth through sixteenth pixels of the first row. What this means is that you have to go through a little arithmetic to find the correct bit to change in hi-res memory, given X (in the range of 0 to 319) and Y (in the range of 0 to 199).

Screen memory is used to determine the color of the pixels in the area normally occupied by a character. The high nibble determines the color of all the bits set to 1, and the low nibble determines the color for the 0's.

Multicolor bit-map mode reduces the resolution to 160 by 200. As with multicolor character mode, the bits in hi-res memory are considered in pairs to determine the color of the corresponding double-width pixel on the screen. Combination 00 selects the screen color (background 0), 01 gets the color from the high nibble of the appropriate byte in screen memory, 10 gets the color from the low nibble in screen memory, and 11 gets the color from the 4-bit color memory location.

Commodore plans a VSP Cartridge, which will include convenient commands for high-resolution graphics.

Fine Scrolling

The VIC II chip allows the whole screen to be scrolled up, down, left, or right by only one pixel. To make this work smoothly, there are provisions to reduce the width of the screen to 38 columns and to reduce the height to 24 columns. That allows two columns (and/or one row) to be hidden, while characters are lined up before fine scrolling into the visible area of the screen. The programming for this smooth scrolling is best accomplished with some simple machine-language routines.

Sprites

What is a sprite? The name doesn't really mean much, but the concept is similar to "Player/Missile Graphics" on Atari computers. Each sprite is a high-resolution entity, 24 by 21 pixels, maintained by the VIC II chip. To program one all you need to do is define its bit pattern, select its color, select its X-Y position, and turn it on. By changing the X and Y values you can move the sprite to any position on (or off) the screen.

Now, for the details... Eight sprites may be displayed on the screen at one time. Each sprite has a one-byte pointer at the top of the screen RAM block. The pointer indicates a 64-byte block within the 16K bank currently selected for the VIC II. The last byte of the 64 is a control byte; the others contain the pixel data for the screen representation of the sprite. Each three bytes represent a 24-pixel row in the sprite. In the standard mode, a bit set to 1 displays a pixel of the selected color and a bit set to 0 displays what's under it (usually the background, but it could be part of a sprite of lower priority!).

Associated with each sprite are several other memory locations in the VIC II chip. The sprite display enable register has a bit for each sprite, as do the sprite multicolor enable, sprite expand 2X horizontal, sprite expand 2X vertical, sprite-to-background priority, sprite-to-sprite collision detect, and sprite-to-background registers. Also, there is a byte for each sprite's vertical position, and a byte for each sprite's horizontal position. Since there are more than 256 possible horizontal positions, there is also a byte containing a ninth X-position bit for each sprite. It sounds — and is — complicated. However, this complexity is required to maintain such a powerful graphics mode. Read on for details of the different capabilities of sprite graphics.

Standard sprites can be displayed in any one of the sixteen colors in a resolution equivalent to the standard bit-map mode. Multicolor mode allows up to four colors in each sprite, and the colors are determined by considering bit pairs in the sprite definition. 00 selects screen color, 01 the color in sprite multicolor register #0, 10 the color in the appropriate sprite's color register, and 11 the color in sprite

multicolor register #1. As with the other multicolor modes, the horizontal resolution is decreased and the sprites are displayed using double-width pixels.

Each sprite can be expanded to double its horizontal or vertical dimension or both.

To handle smoothly the entry and exit of sprites on the screen, the possible X and Y positions actually extend beyond the visible portion of the screen. That way it is possible to have a corner or an edge appear first, followed smoothly by the rest of the sprite.

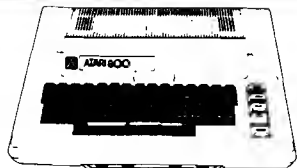
I mentioned priorities earlier. The sprites themselves have fixed priorities with respect to each other: sprite 0 is higher priority than sprite 1, 1 higher than 2, and so on. However, each sprite may be selected to be higher or lower in priority with respect to the background data. Objects of higher priority will overwrite objects of lower priority.

Collisions are detected by the VIC II and appropriate bits are set in two registers. If the corresponding sprite is involved in a collision, then its bit will be set in the register. The bits in the register will remain set until the register's contents are read by your program. Then the whole register is cleared. There is one register for sprite-to-sprite collisions and another for sprite-to-background collisions.

Some of the limitations can be circumvented with more sophisticated programming. For instance, it is possible to display more than eight sprites at once using raster interrupt techniques. Also, because there is so much memory, you can have lots of sprite definitions stored and only alter the pointers. If the fixed sprite priorities are a problem, just swap the pointers and the appropriate bits and registers.

The *Programmer's Reference Manual* gives all the details of the various graphic modes, along with sample programs. Even the little quirks of the system (and ways to get around them) are mentioned. It is good to see Commodore finally paying attention to quality documentation with the VIC-20 and Commodore 64 *Programmer's Reference Guides*. The Guide for the 64 should be available in early December.

MICRO™



400
16K..... \$269
32K..... \$349
48K..... \$429

410 Recorder.....	\$ 76.00
810 Disk Drive.....	\$449.00
822 Printer.....	\$269.00
825 Printer.....	\$589.00
830 Modem.....	\$159.00
820 Printer.....	\$259.00
850 Interface.....	\$169.00
CX40 Joysticks (Pair).....	\$ 18.00
CX853 Atari 16K Ram.....	\$ 77.95

800 — 48K
~~\$369~~ call

Microtek 16K Rem.....	\$ 74.95
Axon Ramdisk (128K).....	\$429.95
Intec 48K Board.....	\$159.00
Intek 32K Board.....	\$ 74.00
One Year Extended Warranty.....	\$ 70.00
CX481 Entertainer Package.....	\$ 69.00
CX482 Educator Package.....	\$130.00
CX 483 Programmer Package.....	\$54.00
CX 484 Communicator Package.....	\$344.00
Atari 800 Dust Cover.....	\$6.99
Atari 400 Dust Cover.....	\$6.99
Atari 810 Dust Cover.....	\$6.99

PERCOM

Disk Drives For Atari Computers

S1 Single Drive.....	\$589.00
A1 Add-on Drive.....	\$339.00
S2 Dual Drive.....	\$879.00
Single Side Dual Head.....	\$679.00
Dual Drive Dual Head.....	\$1046.00

μ-SCI

MICRO-SCI

Disk Drives For Franklin & Apple

A2.....	\$319.00
A40.....	\$369.00
A70.....	\$499.00
C2 Controller.....	\$79.00
C47 Controller.....	\$89.00

ATARI

Pac Man.....	\$35.00
Centipede.....	\$35.00
Caverns of Mars.....	\$32.00
Asteroids.....	\$29.00
Missile Command.....	\$29.00
Star Raiders.....	\$35.00

DATASOFT

Pacific Coast Highway.....	\$25.00
Canyon Climber.....	\$25.00
Tumble Bugs.....	\$25.00
Shooting Arcade.....	\$25.00
Clowns and Balloons.....	\$25.00
Graphic Master.....	\$30.00
Graphic Generator.....	\$13.00
Micro Painter.....	\$25.00
Text Wizard.....	\$89.00
Spell Wizard.....	\$64.00
Bishop's Square.....	\$25.00

ON-LINE

Jawbreaker.....	\$27.00
Softporn.....	\$27.00
Wizard and the Princess.....	\$29.00
The Next Step.....	\$34.00
Mission Asteroid.....	\$22.00
Mouskattack.....	\$31.00

SYNAPSE

File Manager 800.....	\$79.00
Chicken.....	\$26.00
Dodge Racer.....	\$26.00
Synassembler.....	\$30.00
Page 6.....	\$19.00
Shamus.....	\$26.00
Protector.....	\$26.00
Nautilus.....	\$26.00
Slime.....	\$26.00
Disk Manager.....	\$24.00

K-BYTE

Krazy Shoot Out.....	\$32.00
Krazy Kritters.....	\$32.00
Krazy Antics.....	\$32.00
K-star Patrol.....	\$32.00

STICK STAND
\$6⁹⁹



VISICORP

For Apple, IBM, Franklin

Visidex.....	\$189.00
Visifile.....	\$189.00
Visiplot.....	\$159.00
Visiterm.....	\$189.00
Visitrend/Plot.....	\$229.00
VisiSchedule.....	\$229.00
Desktop Plan.....	\$189.00
VISICALC.....	\$179.00
for Apple II plus, Atari, CBM & IBM	

Continental

The Home Accountant (Apple/Franklin).....	\$59.00
The Home Accountant (IBM).....	\$119.00
1st Class Mail.....	\$59.00

FLOPPY DISKS Maxell

MD I (Box of 10).....	\$36.00
MD II (Box of 10).....	\$46.00
MFD I (8").....	\$44.00
MFD II (8" Double Density).....	\$54.00

Verbatim

5 1/4" SS DD.....	\$26.00
5 1/4" DS DD.....	\$36.00

Elephant

5 1/4" SS DD.....	\$19.99
-------------------	---------

TIMEX

TIMEX SINCLAIR 1000

**LOWEST
PRICE
EVER!**

\$89.00

NORTH STAR ALTOS

Call for price and availability
on all models.

MODEMS Hayes

Smart.....	\$239.00
Smart 1200 (1200 baud).....	\$549.00
Chronograph.....	\$199.00
Microdem II.....	\$279.00
Microdem 100.....	\$309.00

Novation

Cat.....	\$144.00
D-Cat.....	\$159.00
Auto Cat.....	\$219.00
212 Auto Cat.....	\$589.00
Apple Cat II.....	\$339.00
212 Apple Cat II.....	\$609.00

Anchor

Merk I (RS-232).....	\$79.00
Mark II (Atari).....	\$79.00
Mark III (TI-99).....	\$109.00
Mark IV (CBM/PET).....	\$125.00
Mark V (OSBORNE).....	\$95.00
Mark VI (IBM-PC).....	\$179.00
Mark VII (Auto Answer/Dial).....	\$119.00
9 Volt Power Supply.....	\$9.00

MONITORS AMDEK

300G.....	\$169.00
Color I.....	\$339.00
Color II.....	\$699.00
Color III.....	\$429.00

BMC

12" Green.....	\$ 85.00
13" Color 1400.....	\$279.00
13" Color 1401 (Mid Res).....	\$369.00

ZENITH

ZVM 121.....	\$ 99.00
--------------	----------

SHARP

Sharp 13" Color TV.....	\$275.00
-------------------------	----------

PANASONIC

TR-120 MIP (High Res. Green).....	\$159.00
CT-160 Dual Mode Color.....	\$299.00

west

800-648-3311

west

IN NV. CALL (702) 588-5854

P.O. BOX 8889 STATELINE, NV. 89449

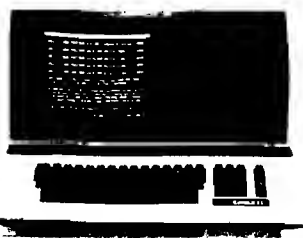
In-stock items shipped same day you call. No risk, no deposit on C.O.D. orders. Pre-paid orders receive free shipping within the continental United States with no waiting period for certified checks or money orders. Add 3% (minimum \$3.00) shipping and handling on all C.O.D. and Credit Card orders. NV and PA residents add sales tax. All items subject to availability and price change. **NOTE:** We stock manufacturer's end third party software for most all computers on the market! Call today for our new catalogue.

FRANKLIN ACE 1000



64K Personal Computer
Hardware, Software and peripheral compatible with the Apple II and even has some features not found on the Apple.

EAGLE COMPUTER



64K RAM
780 KB Disk Storage
Word Processing, Ultracalc CP/M,
CBasic Software
Smith Corona TP1
Letter Quality Printer **Our Price**
Retail Value \$4895.00 **\$2995.00**

commodore

8032	\$999.00
CBM 64	CALL
4032	\$749.00
8096 Upgrade Kit	\$369.00
Super Pet	\$1599.00
2031	\$369.00
8250 Double Sided Disk Drive	\$1699.00
D9060 5 Megabyte Hard Disk	\$2399.00
D9090 7.5 Megabyte Hard Disk	\$2699.00
8050	\$1299.00
4040	\$969.00
8300 (Letter Quality)	\$1549.00
8023	\$599.00
4022	\$399.00
New Z-Ram, Adds CP/M and 64K Ram	\$549.00
The Manager	\$209.00
Magis	CALL
Word Pro 5 plus	\$319.00
Word Pro 4 plus	\$299.00
Word Pro 3 plus	\$199.00
The Administrator	\$379.00
InfoPro Plus	\$219.00
Power	\$79.00
VIC 20 Dust Cover	\$6.99
CBM 8032 Dust Cover	\$14.99
CBM 8050/4040 Dust Cover	\$10.99

HEWLETT PACKARD



HP-85
\$1969



HP-125	\$1969.00
HP-85 16K Memory Module	\$169.00
5 1/4" Dual Master Disk Drive	\$1799.00
Hard Disk w/ Floppy	\$4349.00
Hard Disk	\$3549.00
"Sweet Lips" Plotter	\$1199.00
80 Column Printer	\$649.00

IBM®

PC



Call...
for price and availability
on IBM-PC hardware, soft
ware and peripherals.

NEC 3550 Printer (for IBM) \$2099.00

VIC 20
\$179



VIC 1530 Commodore Datasette	\$69.00
BIC 1540 Disk Drive	\$339.00
VIC 1541 (64 Disk Drive)	CALL
VIC 1525 Graphic Printer	\$339.00
VIC 1210 3K Memory Expander	\$32.00
VIC 1110 8K Memory Expander	\$53.00
16K VIC Expansion	\$94.00
VIC 1011 RS232C Terminal Interface	\$43.00
VIC 1112 VIC IEEE-488 Interface	\$86.00
VIC 1211 VIC 20 Super Expander	\$53.00
VIC Mother Board	\$99.00



**HEWLETT
PACKARD**
HP 41CV
CALCULATOR

\$209



HP 41C	\$149.00
HP 10C	\$69.00
HP 11C	\$79.00
HP 12C	\$114.00
NEW 115C	\$109.00
NEW 16C	\$114.00

HPIL PERIPHERALS IN STOCK!

Televideo Terminals

910	\$579.00
912C	\$699.00
920C	\$749.00
925C	\$749.00
950	\$950.00



800A	\$1319.00
802	\$2649.00
802H	\$4695.00
806	\$5795.00
816	\$9495.00

PRINTERS

Smith-Corona

TP-1

\$599



C.I.TOH (TEC)

Starwriter (F10-40CPS)	\$1399.00
Printmaster (F10-55CPS)	\$1749.00
Prowriter 80 Col (P)	\$499.00
Prowriter 80 Col (S)	\$629.00
Prowriter 2 (132 Col)	\$799.00

Okidata

82A	\$429.00
83A	\$659.00
84P	\$1079.00
84S	\$1199.00

IDS

132 (fully configured)	\$1599.00
80 (fully configured)	\$1399.00

Call for other configurations.

Daisywriter

Letter Quality	\$1049.00
----------------	-----------



PC-1500
POCKET COMPUTER

ALSO AVAILABLE:
Printer w/cassette interface
cassette tape recorder
and 4K and 8K RAM EXTENSIONS

NEC COMPUTERS

8001-A	\$749.00
8031	\$749.00
8012	\$549.00

Printers

8023	\$549.00
7710/7730	\$2399.00
3510/3530	\$1599.00

Monitors

JB-1201	\$159.00
JC-1201	\$329.00
JC-1202	\$899.00

east

800-233-8950

IN PA. CALL (717) 327-8575

477 E. THIRD ST., WILLIAMSPORT, PA. 17701

In-stock items shipped same day you call. No risk, no deposit on C.O.D. orders. Pre-paid orders receive free shipping within the continental United States with no waiting period for certified checks or money orders. Add 3% (minimum \$3.00) shipping and handling on all C.O.D. and Credit Card orders. NV and PA residents add sales tax. All items subject to availability and price change. **NOTE:** We stock manufacturer's and third party software for most all computers on the market! Call today for our new catalogue.

computer mail order east

Command-O or Command-O-Pro?

It's called
in the U.S.

Command-O or Command-O-Pro?

20 additional commands for editing and debugging text commands (And our technical writer dug out the commands in the course of doing a 76 page manual.)

Extends Commodore's 8032 advanced screen editing features to the ultimate. **SCROLL** up and down, insert or delete entire lines, delete the characters to the left of the cursor select **TEXT** or **GRAPHICS** mode, change the PET's bell, Redefine the window to adjust its size and position on your screen. And define any key to equal a sequence of up to 99 key strokes.

Resides in hexadecimal address \$0040. The rightmost 4 bits of the address are available at the 4041 socket in 4016 and 4032 of the rightmost 8032.

Electric Works
Inc.

Command-O from Skyles Electric Works
Complete with Socket 2ME (2 sockets)
Complete with GO-4IT (4 sockets)

Disk-O-Pro ROM:

Disk-O-Pro ROM

5 new commands at \$75.00 a command!

The powerful chip for all "basic" PCs is the Version III (BASIC 2.0) ROMs. The Commodore Disk-O-Pro will give you software containing 15 instructions - 15 identical with the new BASIC 2.0 with 8032 ROMs) compatible with both old and new Plus 4 additional disk commands - including MERGE and appending MERGE program overlaying and adding and PRINT USING, allowing formatting numbers and numbers on the P.C.T. screen or on paper.

NOTE: Old DOS doesn't recognize 2 commands - MERGE and PRINT USING (SET which allows you to control reference point to BASIC DOS scrolling and SET which allows you to control...

NOTE: On the PCT screen, if you want to print a sequence of numbers, use the **PRINT USING** command. For example, to print numbers from 1 to 10, use the command **PRINT USING "###" 1 TO 10**. The **PRINT USING** command is described in the **PRINT USING** command reference page.

to ensure that the
are all the same
extra editing features
music on your PET
Completely compatible with
ToolKit, the chip resides in the same slot as the
address 30000, the highest address available to
PETs. And for the owners of Command O's, the
do have interface built
For those owning a BASIC 2.0, the 6502 even handles the
Disk II Pro may not be suitable, the Command O's
have never abandoned a PET owner.
Complete with a page manual written by Greg Job
Program Shires Electric Works
A Interface Box of the 6502 - BASIC PETs
on graphics are 65

Complete with 26-page manual.

Qse-O-Program Styles Electric Work

Complete with Interface Board for "Classic" PC

For full-resolution graphics see GRAPHIX

WORD WRITER

WORD WRITER

- And this one small program can help increase the power of your IBM® Model 5150.
- This terminal device can help make business writing the easiest thing in the world.
- One more task you can do better than ever before.
- You'll love it! Call 800-675-1515 or write us today.

[illegible][illegible]

No more typing out or printing out
your letters and you can print out
your letters and you can print out
your letters and you can print out

Call 800-363-9886 or Super Pet SP93000 \$85.00

WORD WATER

Please specify ink/PEN/CBM model when ordering.

For a full range of word processing products see

In the like

"The PET with the new PicChip... is like home movies."

The PET with a new PicChip... is a home movie.

On Skyline, a ROM chip, the BASIC 1.0 or BASIC 2.1, BASIC 4.0 or BASIC 4.1, and the commands that allow you to control the dynamic graphics displays: plots, bars, and routing, scrolling, shifting and inverting, and easily added to your BASIC program.

The address for the 20000000 (which would require a PICChip module PC2) for the 4000 (PC4) and for the 8000 (PC8) is 4000, unless you have a Mikro, Word Pro 3 or J/Visum, you will need the same address. In those cases, you will need the PicChip on an interface multi-board. In all cases, the Mikro, WordPro or SOFTS would be a wonderful addition to the Skyline Socket 2ME or GO-4IT available at a very special price.

	\$60.00
	\$75.00
	\$90.00

PicChip from Skyline Electric Works
Complete with Socket 2ME (2 sockets)
Complete with GO-4IT (4 sockets)

Skyline Catalogue Page

Skyles Catalogue Page

For PET and CBM owners only:

This is just 1 of 20 pages of the newest and biggest Skyles catalog, hot off the press.

We know you'll want this page, in its full 8½ x 10 splendor, and another 19 pages of peripherals, software and books that will make your PET or CBM computer even nicer to live with. So, if we missed sending

you your very own
copy within the last
few weeks, call us at
(800) 227-9998
(unless you live
in California, in
which case call
(415) 965-1735.

From Skyles
Electric Works, the
oldest and largest
professional
specialists in the
business.



Skyles Electric Works
231-E S. Whisman Road
Mountain View, California 94041

Microcomputers in a College Teaching Laboratory, Part 2

by Richard Heist, Thor Olsen, and Howard Saltsburg

Many laboratory situations involve measuring continuous ranges of light, heat, and sound. An inexpensive device to help the digital computer deal with these analog quantities is the analog transducer. Specific applications to temperature and light intensity measurement are discussed.

Part 1 of this series (MICRO 53:53)¹ gave an overview of the microcomputer laboratory program at the University of Rochester, Department of Chemical Engineering. In this article the problems of measuring physical, chemical, and mechanical properties will be addressed, since such problems are common to most engineering and scientific laboratories. Temperature, pressure, flow, and light intensity are typical quantities of interest, and in many cases the required information is provided by a transducer in the form of an analog signal, usually electrical in nature. Difficulties in the measurement and conversion to the desired physical or chemical quantity of these signals may tend to obscure the purpose of the measurement. The microcomputer often offers a simpler alternative to more conventional laboratory instrumentation, thus making it easier for the user to maintain a focus on the purpose of the measurements. Furthermore, it combines this decrease in complexity with low cost, high speed, reliability, and precision.

In what follows, the use of simple interfacing devices will be discussed. These devices were selected for their flexible operating characteristics, which give them quite general utility. Examples will illustrate their application to the measurement of temperature and light intensity. The emphasis will be on specific applications, not on

design or construction of the devices, which are very simple.

Analog Signals and A/D Converters

When the transducer of interest produces an electrical signal, the problem of property measurement is reduced to one of measuring that signal (usually voltage, current, or resistance) to the desired degree of accuracy and at an appropriate rate. Many laboratory measurements require only slow (< 50 Hz) data acquisition rates or low (8-bit) precision. The actual requirements should be evaluated carefully and realistically since they have an important bearing on the technique and instrumentation used to measure the electrical quantities.

When high-speed data acquisition and high resolution are not needed, it is remarkably easy to interface many laboratory experiments and measuring devices to the computer. As will be demonstrated, an appropriate A/D converter, selected for its flexibility, combined with a microcomputer and a high-resolution dot matrix printer, becomes a versatile data acquisition system [the universal instrument referred to in the first article in this series (MICRO 53:53)]. This combination can be used effectively and inexpensively to solve many laboratory measurement problems.

The two types of A/D converters, which have been widely used in the Rochester program, both employ a pulse-width technique for data conversion, even though one is used to measure voltage and the other resistance. Each device, upon command from the computer (a trigger pulse) begins a timing cycle, the length of which is proportional to the magnitude of the applied analog signal. At the end of the cycle, the converter signals the

computer that conversion is complete (end of conversion, EOC).

The computer is programmed to measure the length of the timing cycle by repeatedly incrementing the microprocessor index registers until the EOC signal is received. The microprocessor requires a fixed number of machine cycles to run through the program loop in which it tests for EOC and increments the index registers. Since these cycles are accurately timed by the internal crystal oscillator, the count accumulated in the index registers is proportional to the elapsed time. By suitable calibration, this count can be converted to the desired data format, and the measurement is complete.

Typical resolution can range from eight to 12 bits; the corresponding conversion times are approximately three to 200 milliseconds. The ability to trade off conversion time for resolution gives these simple devices a flexibility not shared by other kinds of A/D converters and makes them feasible for many laboratory applications.

The device used for voltage measurements is a QM-100 A/D converter (Analog Systems, P.O. Box 35879, Tucson AZ). This device has three independent A/D channels, each with a 0 to 10 VDC input range. In operation, a voltage ramp generator is triggered by the computer, and its output is compared to the transducer voltage. A comparator signals the computer when the ramp just exceeds the transducer voltage (EOC).

For resistance measurements, a simple A/D method outlined in an article in MICRO² was chosen. It uses a 555 timer IC in the configuration shown in figure 1. The conversion method involves charging the timing capacitor, C1, to a fixed voltage through the transducer resistance, R, and measuring the charging time with

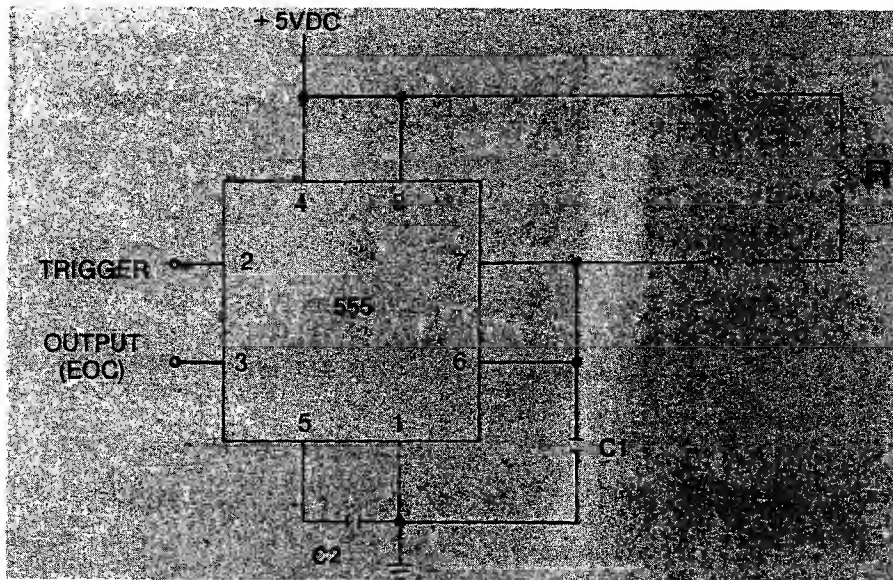


Figure 1: A 555 timer integrated circuit wired as a monostable multivibrator. A typical value for C2 is .01 μ F. The value chosen for C1 depends upon R. For instance, if R = 150 K Ω and 10-bit conversion is desired (1024 counts, see text), then C1 should be about 0.1 μ F (see reference 3).

the computer. The computer triggers the charging process and is then signaled by the 555 timer when conversion is complete. By choosing the appropriate combination of transducer and timing capacitor for a specific application³, you have a simple and inexpensive data acquisition system.

While the examples described here are specific to temperature and light-intensity measurements, the concepts are general. These interfacing methods can be extended to virtually any kind of voltage or resistance measurement. Moreover, it is clear that the use of a resistance transducer, when appropriate, can result in a significant simplification of hardware, compared to other techniques, and it will often pay to change to sensors of this type.

One additional point that should be made in connection with the pulse-width A/D converters is the ease with which these devices can be multiplexed. Many times it is necessary to measure a number of inputs simultaneously. Since most microcomputers will support only a limited number of I/O lines, it is useful to be able to switch-select devices automatically (multiplex). Examples of this include the simultaneous monitoring of the temperature of each tray of a multistage distillation column and multiple concentration profile measurements along a tubular reactor. The circuit shown in figure 2 has been used to multiplex the sensors in several experiments. It is based on the 74150 IC, a 16-channel

multiplexer. A similar circuit, based on the 75151 IC, can be used to construct an 8-channel device. Both multiplexer ICs and their operation are described in detail in the literature listed in reference 4.

Construction details have not been discussed at length since they are adequately described in the microcomputer and electronics literature⁵, but good construction techniques must not be underemphasized, particularly for applications requiring higher precision. The important construction practices are documented in the literature and are well known to experienced personnel. Do not hesitate to ask for advice.

Some care should be exercised in the use of the converters. For instance, the characteristics of all electronic components are, to some extent, temperature-dependent. Therefore, large fluctuations in ambient temperature should be avoided during data collection or between calibration and actual use. Another point concerns the use of the 555-based converter in the triggered mode described above. When the EOC is reached, the 555 IC starts discharging the timing capacitor and the system will remain in discharge mode until it is triggered again. If the time between EOC and the next trigger pulse varies, the circuit may operate with varying levels of residual charge on the timing capacitor. The result will be timing er-

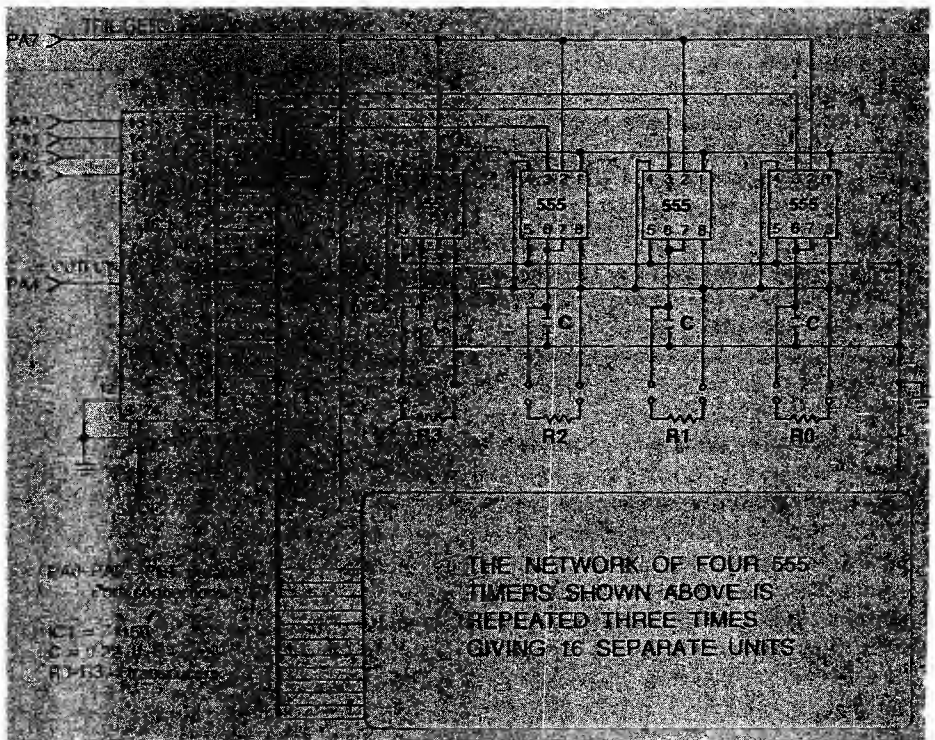


Figure 2: A 16-channel multiplexer circuit based on a 74150 TTL Integrated circuit. The end-of-conversion signal, pin 3, of any of the 555 timers can be accessed by placing the appropriate binary number (0-15) on the input pins (15, 14, 13, and 11, respectively) of the 74150. In the diagram, PA0 - PA4 and PA7 represent PET parallel port connections. The output from the 74150 is available at pin 10. The resistance value of the transducers, R0 - R15, will determine the value of the charging capacitor, C (see figure 1). A typical value is $0.22 \mu\text{F}$ (see reference 3).

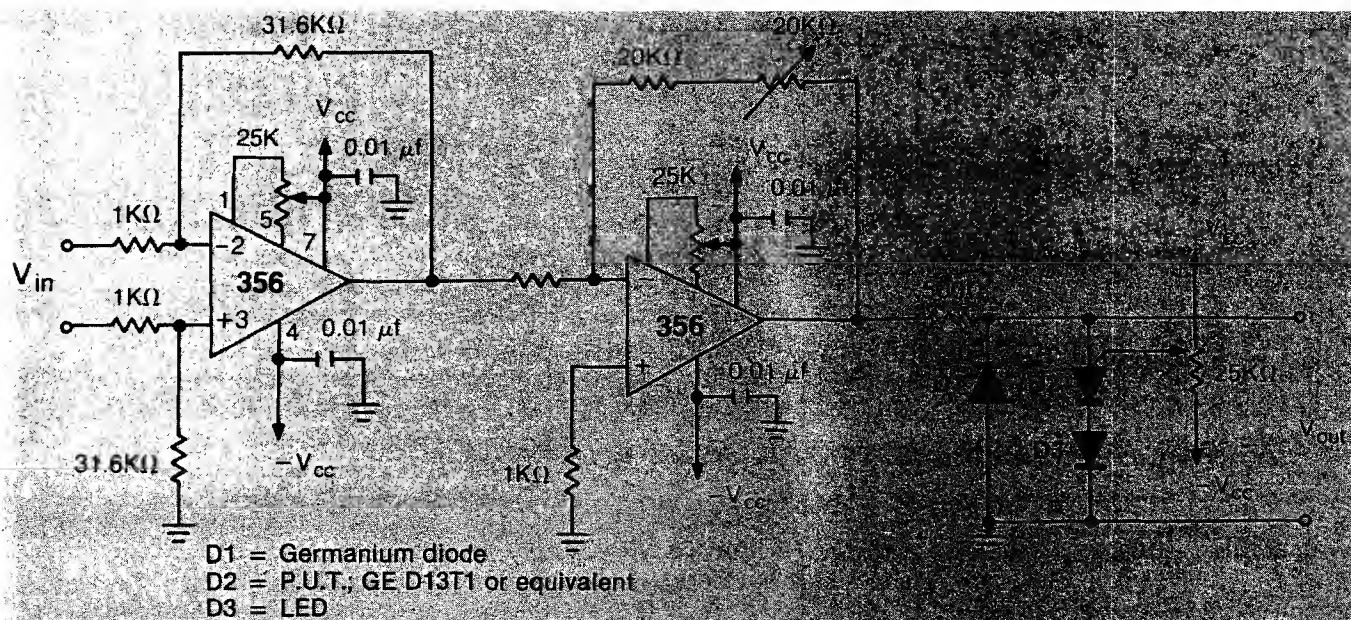


Figure 3: A two-stage voltage amplifier. The overall gain ranges from 630 to 1260, depending upon the setting of the 20 KΩ variable resistor in the feedback loop of the second stage. The optional diode network ensures that the output voltage will be positive (D1) and will not exceed 10VDC (D2). This is a requirement for proper operation of the QM-100 A/D converter. D3 is used to indicate over-ranging.

rors, leading to poor reproduction of the data. The problem can be circumvented by introducing a sufficient delay between measurements to assure total discharge, or by operating the system with reproducible discharge time.

Temperature Measurement

Two analog electrical signals commonly associated with temperature are thermocouple voltage and thermistor resistance. The problem is to provide a convenient method for measuring these analog signals, then convert the results to temperature.

Consider, for example, a temperature measurement in which a precision of one degree Celsius is desired at a temperature of 100 degrees. If the sensor is a thermocouple, the transducer output will be in the low millivolt range and a difference of one degree in temperature would produce a voltage difference of, at most, a few tens of microvolts — beyond the direct resolution of most analog meters. As the precision requirement of an experiment increases, conventional thermocouple instrumentation becomes costly.

With digital instrumentation, this precision is not difficult to achieve. Provided the input signal at 100 degrees is within the upper half of the converter's input range, all that is required is an eight-bit A/D converter. An obvious problem, then, in interfacing thermocouples [and many other laboratory devices as well] is the low level of

the output voltages. The millivolt-level signals generally available must be amplified to the 0.5 to 10 VDC range before A/D conversion can be performed satisfactorily. Fortunately, the frequency response requirements are minimal for most applications, so large-gain amplifiers (100X - 2000X) are relatively simple to build⁶. See figure 3 for a typical example. When adjustable gain is included, the combination amplifier and QM-100 converter becomes an A/D system that is inexpensive, versatile, and reliable.

Thermistors, in contrast to thermocouples, can be manufactured to provide large resistance changes for small temperature differences. Unfortunately, the response is highly non-linear, and the response characteristics tend to be non-uniform, even among thermistors of the same kind. These properties make it difficult and expensive to reduce thermistor output to temperature with analog hardware. Using a microcomputer with the 555 timer A/D, on the other hand, you can easily handle these complex relationships with appropriate software modifications.

Light-Intensity Measurement

Another property commonly measured in laboratories is light intensity. In chemical laboratories, this measurement is usually made with commercially available instrumentation equipped with photocells or photomultiplier

tubes (e.g., colorimeters and spectrophotometers). It has proven to be easy to use either the QM-100 or the 555 converter to interface the microcomputer to such optical instruments. In fact, inexpensive colorimeters based on a 555 timer/photoresistor circuit can be built to almost any geometry required by an intended application.

For photomultiplier-equipped spectrophotometers where the output signal is a current, a simple circuit can be used to convert the transducer output to a voltage⁶. A typical example of a current-to-voltage converter circuit is shown in figure 4. Once a voltage is available, the procedure for using the QM-100 is the same as described above.

A major use of this type of optical instrumentation is in measuring the concentration of light-absorbing chemicals in liquids and gases. Normally, the response of such instruments is proportional to the inverse exponential function of the concentration. Thus, should a linear response be required when using a chart recorder for data acquisition, an expensive linearizing module must be added.

In some cases, not only is a linear response required, but the quantity of interest is the total amount of a chemical that has passed through the detector. This type of measurement requires the capability to integrate a response over time — another module to add to the recorder.

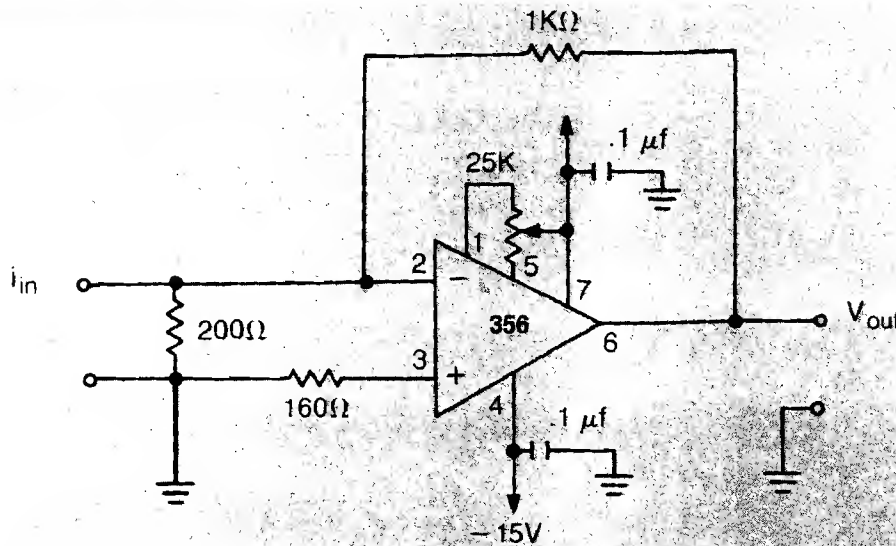


Figure 4: A current to voltage converter. The circuit shown here will typically produce millivolt-level output for microampere-level input with good frequency response.


When the microcomputer is used to monitor such instruments, these conversions require only a few lines of additional code in the applications program. Within the limits of the microcomputer's capabilities, any relationship between sensor output and the quantity of interest can be accommo-

dated without additional cost as long as the relationship can be adequately described by mathematical expressions. Also, since the computer can store spectral data between scans, it is possible through computer interfacing to convert a single-beam spectrophotometer into a pseudo dual-beam device.

The simplicity of microcomputer-based systems can best be illustrated by the measurement of optical density of fluids. An extremely simple colorimeter, useful for many chemical concentration measurements, can be constructed from a suitable light source, such as a light-emitting diode, and a photoresistor, placed on opposite sides of a translucent vessel containing the fluid to be studied. The photoresistor is interfaced via the 555 A/D converter. Since the components (light source and photoresistor) can be very small, e.g. three mm diameter, and the units are so simple, a variety of geometries can be accommodated. Thus, a chemical reaction involving a color change can be followed *in situ* in a small test tube. There is no need to disturb the process by withdrawing samples for analysis.

Another example is the study of the dispersion of a dye in a liquid flowing in a long tube. It is a simple matter to place these LED-photoresistor colorimeters in collars clamped around the tube, at intervals, and observe the dispersion effect without disturbing the flow.

Note that when a LED is used in



**PREMIER
ISSUE**

DECEMBER 1, 1982

Commander

**THE MONTHLY JOURNAL FOR
COMMODORE
COMPUTER USERS**

VIC-20 64 PET/CBM
MAX MACHINE

"COMMANDER will be dedicated to communicating the fun of, as well as the latest information about the COMMODORE COMPUTERS."

GET YOUR MONEY'S WORTH

You've probably made a sizeable investment in your computer equipment. COMMANDER can help you make the most of it. Each issue brings you the no-nonsense advice you need to stay on the leading edge of this constantly changing field. COMMANDER will be your reference source to the world of computers with the best, most comprehensive coverage you can get!!

PREMIER ISSUE ☐ 1 YR. \$22 ☐ 2 YR. \$40 ☐ 3 YR. \$56
(PRICES DO NOT INCLUDE \$4 DISCOUNT)

\$4 DISCOUNT

— Subscription Orders Only —
Toll Free Number: 1-800-426-1830
(except WA, HI, AK)

COMMANDER
P.O. BOX 98827
TACOMA, WASHINGTON 98498
(206) 565-6818

SOPHISTICATED TELE-COMMUNICATION IS HERE

THE COMMUNICATOR

for 4.0 Commodore Computers

With: **HIGHER SPEED**
MORE SOPHISTICATED CONTROL
LOWER PRICE

THE HARDWARE — A printed circuit board; easily installed in the CBM. It uses no CBM connectors; gives a serial port with true RS232C standard. The board alone is capable of running up to 9600 baud. With the software it will run up to 4800 baud.

THE SOFTWARE —

- Emulates the ADDS Regent 100, ADM 31 and/or the TeleVideo 950.¹ Or choose the VT100 model for use with DEC and VAX computers.
- Runs coresident with BASIC programs; lets BASIC programs and program on host computer communicate to develop really sophisticated communication and control capabilities.
- The program is on ROM at either address; no disk loading required. Uses only 512 bytes of RAM; will relocate itself around any other machine language program at top of memory.
- Will upload and download and run BASIC programs. With BASIC program will upload and download standard data files. 100 page manual gives program listing for BASIC programs.

Excellent text editor designed to work with THE COMMUNICATOR

THE COMMUNICATOR	\$200	THE COMMUNICATOR and	
Text Editor	\$ 40	U.D.S. 1200 baud modem	\$625
U.D.S. 1200 baud	\$450	THE COMMUNICATOR and Hayes	
Hayes 300/1200 baud	\$595	300/1200 baud modem	\$770

We sell U.D.S. and Hayes 300 baud modems at excellent prices

AMPLIFY, INC.

2325 Macbride, Iowa City, Iowa 52240 319-337-8378

¹trademarks Adds Regent, Inc., Lear Siegler, Inc., Televideo Systems, Inc.

this mode it is important that it is supplied a constant current. A simple circuit that will accomplish this⁷ is shown in figure 5.

Concluding Comments

The general utility of the A/D converter (computer) printer combination deserves reiteration. In going from one application to another, only portions of the applications program need to be changed; the data acquisition routines remain unaltered. The A/D devices previously described can be adapted to a variety of resistance, voltage, and current measurements with little or no modification. The flexibility of these A/D converters, the computational capability of the microcomputer in the reduction of data, and the high-resolution hard copy capability of the dot-matrix printer are combined to make the system an inexpensive but powerful universal data acquisition instrument.

Once it is realized that resistance and voltage can be measured so easily with the microcomputer, you may wish to redesign existing experiments to match the output to the interface, rather than the other way around. In particular, it may be advantageous to generate resistance, rather than current or low-level voltage; e.g., use thermistors instead of thermocouples.

At moderate expense, the system can be expanded further to provide the capability to feed back information and change the operating conditions of the device it monitors. Digital to analog conversion and control will be discussed in a subsequent paper.

The role of the computer in the laboratory is that of a tool. Certainly it is a remarkable tool in terms of power and capability; but nevertheless, it is a means to an end and not the end in itself. This point is sometimes too easily forgotten.

References

1. H. Saltsburg, R.H. Heist, and T. Olsen, *MICRO, The 6502/6809 Journal*, [53:53].
2. J. Sherburne, *MICRO, The 6502/6809 Journal*, [26:31].
3. See, for example, H. Berlin, "The 555 Timer Applications Sourcebook with Experiments," (Howard W. Sams and Co. Inc., Indianapolis, 1979).
4. See, for example, "Signetics Logic — TTL Data Manual," (Uniplan, San

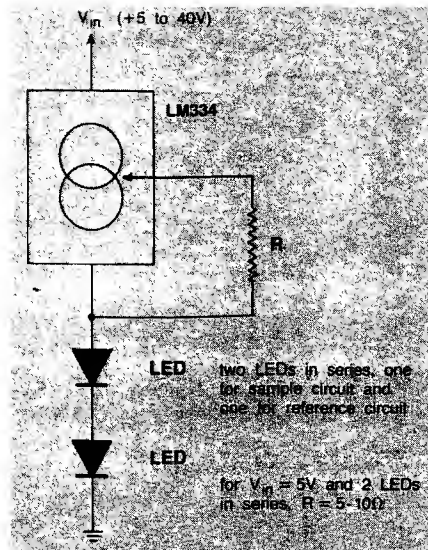


Figure 5: A current regulator. The LM334 is an adjustable current source with good current regulation. A typical value for R with two LEDs in series is 5 to 10 ohms. The two LEDs in series are used to provide a sample signal and a reference signal for the colorimeter applications discussed in the text.

- Francisco, 1978); "The TTL Data Book," (Texas Instrument, Inc., 1976), 2nd ed.; D. Lancaster, "The TTL Cookbook," (Howard W. Sams and Co. Inc., Indianapolis, 1979).
5. See, for example, P. Horowitz and W. Hill, "The Art of Electronics," (Cambridge University Press, Cambridge, 1980); F.M. Mims, "Engineer's Notebook II. Integrated Circuit Applications," (Tandy Corporation, 1981); Z.H. Meiksin and P.C. Thackary, "Electronic Design with Off-the-Shelf Integrated Circuits," (Parker Publishing Co. Inc., West Nyack, NY, 1980); S.A. Hoenig, "How to Build and Use Electronic Devices without Frustration, Panic, Mountains of Money or an Engineering Degree," (Little, Brown and Co., Boston, 1980) 2nd ed.
6. See, for example, W. Jung, "IC Op-amp Cookbook," (Howard W. Sams and Co. Inc., Indianapolis, 1979); "Operational Amplifiers: Design and Application," (McGraw-Hill Book Company, New York, 1971), edited by J.G. Graeme, G.E., Tobey, and L.P. Huijsman.
7. F.M. Mims, "Engineer's Notebook II. Integrated Circuit Applications," (Tandy Corporation, 1982) p. 116.

You may contact the authors at the Department of Chemical Engineering, University of Rochester, Rochester, NY 14627.

MICRO

PROFESSIONAL WORD PROCESSOR

- Double Columns
- Right Justification
- Printer Graphics
- Variable Line Space
- Printer Control Code
- Page by Paragraph
- Line Centering
- Shorthand
- Margin Control
- Form Letters

FOR APPLE/PET/CBM

COPY-WRITER

only \$185.00

EXCHANGE DATA w IBM 3740

PEDISK II 877 FLOPPY DISK Systems can now read and write records from IBM "Basic Data Exchange" type diskettes. FILEX software from WILSERVE does all the work! Converts EBCDIC - ASCII.

EXCHANGE System (877/FILEX) \$1295.
 PEDISK 877-1 8" Floppy for PET \$ 995.
 PEDISK 540-1 5" Floppy for PET \$ 595.
 CONTROLLER BOARD w PDOS \$ 229.
 PEDISK II is a high performance floppy disk system designed for the Commodore PET/CBM, Rockwell AIM and Synetek SYM. It features high performance, simple reliable design and IBM format.

SOFTWARE FOR PEDISK II
 COPYWRITER Pro Word Processor \$185.
 MAE Macro Assembler Editor by EHS \$170.
 FLEXFILE II Data Base Manager \$ 80.
 PAPERMATE Word Processor \$ 60.
 DISK UTILITY PACK \$ 25.
 FASTFILE Data Base \$100.
 FILEX IBM Access Routines \$245.
 MENU LOAD \$ 10.
 fullFORTH+ \$100.

Commodore Communicates! COMPACK \$129.

Intelligent Terminal Package including ACIA based interface DB25 cable STCP software

- Remote Telemetry
- Transfer to/from Disk
- Printer Output
- XON-XOFF Control
- User Program Ctrl
- Status Line

\$139 COLOR CHART

AIM/SYM system video display, 64 x 16 characters, 8 colors, plugs into ROM socket, 4K RAM Multiple modes; semi graphics, alpha.

PET/CBM color graphic display, 128 x 192 pixels, generate color bar graphs on one screen with data on main screen. RS170 video color chart. 6847 based video output.

COLOR VIDEO FOR PET/CBM/AIM/SYM

ROMSWITCH - 4 ROMS IN 1

SPACEMAKER \$39.95

Switch 4 ROMs into the same socket. A slide switch activates one of four Electronic controls insure no glitches and allow ROM switching under software control. ROMs can be switched from the keyboard.

fullFORTH+ for APPLE/PET

FULL FIG FORTH implementation plus conditional assembler, floating point, string handling, multi-dimensional arrays and disk virtual memory.

fullFORTH+ from IDPC Co. \$100
 Target Compiler \$ 50

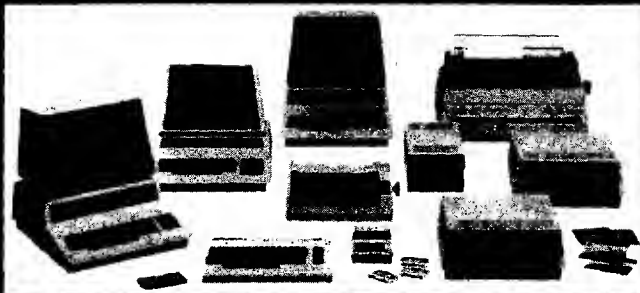
SEE YOUR DEALER OR:

MICROTECH

P.O. Box 102
 Langhorne, Pa. 19047
 215-757-0284

DEALER INQUIRIES INVITED

SJB DISTRIBUTORS. THE MOST COMPETITIVE PRICES ON COMMODORE PRODUCTS.



commodore

INTERFACES

ADA-1450 Serial	\$149
ADA-1600 Parallel	149
RS232 cable for Vic or 64, 2m	30
Video/Audio cable for 64 & monitor	25

MONITORS — Great resolution for the CBM 64 or VIC

Panasonic, 13" Color	\$375
Amdek Color I	330
NEC JB 1201M, 12" Color	330
NEC JB 1201, 12" green phosphor	170
Amdek Video 300L, green phosphor	175

BUSINESS SOFTWARE

Spellmaster Dictionary (great for WordPro!)	\$199
OZZ Data Base System (8050)	240
Silicon Office (database, wp)	995 (New)
Wordcraft 80	289
VisiCalc (new expanded)	199
Dow Jones Portfolio Management System (RS232)	120
WordPro 4+ or 5+	299
The Manager	199
Legal Time Accounting	425
I R M A	295
BPI A/R, G L, Job Cost, Inventory, Payroll	325 pkg

SJB will service any VIC or CBM64.

MasterCard, Visa, Money Order, Bank Check

COD (add \$5) accepted.

Add 3% surcharge for credit cards.

In stock items shipped within 48 hours, F.O.B., Dallas, TX

All products shipped with manufacturer's warranty.

TO ORDER CALL TOLL FREE

800-527-4893 800-442-1048 (Within Texas)

SJB will meet any competitive price under similar in-stock conditions.

SJB DISTRIBUTORS, INC.

10520 Plano Road, Suite 206
Dallas, Texas 75238
(214) 343-1328



Business Hours
M-F 8 to 6
Sat 10 to 2

Prices are subject to change without notice.

SOFTWARE FOR CBM 64

Word Processing	\$90
Computer Tutoring Game (COCO)	50
General Ledger	199
Pet Emulator (emulates 4.0 basic)	30
CBM EasyCalc (for the 64)	99
CBM EasyFinance	50
CBM EasyPlot	80
CBM EasyScan (appointment manager)	80
Sprite-Magic (build sprites on screen with Joystick, save to disk or cassette)	30
Assembler Package for CBM 64 (cassette)	
Editor (creates and updates source code), Assembler, Loader, Disassembler	50
Mail Mate	50
IEEE Interface (64)	100
Parallel Interface	90
RS232 Interface (modems, printers)	45

VIC PRODUCTS

VIC 20 Computer, 5K	\$199
Vic Datasette Recorder	60
Vic 1541 Disk Drive	395
VIC MODEM (for CBM 64)	100
VIC 1525 Graphic Printer (for CBM 64)	325
8K Memory Expansion Cartridge	49
16K RAM	99
24K RAM	155
IEEE Interface (VIC)	85
Gorf (great arcade game)	30
Omega Race	30
Midnight Drive	23
VIC 3 slot Expander	43
VIC 6 slot Expander	83
Seawolf	23
Cosmic Cruncher	23

Arcade Joysticks — Heavy duty with 2 firing buttons! Great for the VIC or 64

	\$25
SuperPET (5 languages, 2 processors)	\$1409
CBM 8032 Computer 80 column	1029
CBM Memory Expansion, 64K	359
PET 4032, 40 Column	950
CBM 8050, 1 Mg. Dual Drive	1259
CBM D9060, 5 Mg. Hard Disk	2240
CBM D9090, 7.5 Mg. Hard Disk	2600
CBM 4040, 340K Dual Drive	919
CBM 2031, 170K Single Drive	489

PRINTERS — LETTER QUALITY

CBM 8300 40cps	\$1450
Diablo 620, 25cps	1350
Nec Spinwriter 7700, 55cps	2350
Nec Spinwriter 3500, 35cps	1600

PRINTERS — DOT MATRIX

CBM 4022, 80cps graphics	\$395
CBM 8023, 150 cps graphics	599
Okidata 82A, 120cps: serial or par.	449
Nec 8023A(parallel)	499

MICROTM

Apple Slices

By Tim Osborn

One of the fastest techniques that lets you search for a specific occurrence of an item within a sorted set is the binary search. This month's column presents a subroutine (BINARY-SEARCH) that you may call from your BASIC programs to perform a binary search on a sorted (ascending) string array. The advantages of a binary search over a serial search increase as the number of items in the array grows. For example, an array of 4096 items can be searched in less than 11 tries.

The Method

A binary search tests the middle element in the remaining part of the array. If the element is higher than the search argument (the value being searched for), the part of the array from this element upward is left out of the search by resetting the upper limit to the index of the element. If the element is lower than the search argument, the part of the array from this element downward is left out by resetting the lower limit to the index of this element. The program then finds the average of the upper limit and the lower limit and searches the element at this location. The procedure continues until the element is found or until it discovers that the upper and lower limits have converged without finding the element.

The Subroutine

The syntax for the binary search is:

& GET (XX\$,YY\$)

where 1. XX\$ represents any legal string array name, and 2. YY\$ represents any legal string variable name. This subroutine will return in SS% the index number of the element in XX\$ that has a value equal to YY\$ if the item is found. If the item is not found the subroutine will return a -1

```

1 ;*****
2 ;*      APPLE SLICES      *
3 ;*  BINARY — SEARCH  *
4 ;*      T. S. O.      *
5 ;*****
6 ;ZERO PAGE EQUATES
7 LOWTR  EPZ $9B          ;WORK POINTER
8 VARNAM  EPZ $81         ;CONTAINS LAST USED VARIABLE NAME
9 VARAD   EPZ $83         ;ADDRESS OF PASSED STRING
10 CHRGET  EPZ $81         ;APPLESOFT'S ROUTINE TO GET A BYTE
11 ;
12 ;EQUATES
13 ;
14 AMPERV  EQU $3F5        ;AMPERSAND VECTOR LOCATED HERE
15 FIND    EQU $E053       ;ROUTINE TO LOCATE VARIABLE DESCRIPTOR
16 CHKOPN  EQU $DEBB       ;CHECK FOR OPEN PAREN
17 GETARYPT EQU $F7D9       ;ROUTINE TO FIND ARRAY DESCRIPTOR
18 CHKCOM  EQU $DEBE       ;CHECK FOR COMMA
19 SYNERR  EQU $DECD       ;DISPLAY SYNTAX ERROR
20 DATA   EQU $D995       ;ADVANCE TEXTPTR TO END OF STATEMENT
21 ;
22 ;
23 ;
24 ;
25 SETVEC  LDA #$4C        ;JUMP ABSOLUTE INSTRUCTION
26 STA AMPERV              ;LSB OF ENTRY ADDRESS
27 LDA #ENTRY              ;
28 STA AMPERV+1            ;MSB OF ENTRY ADDRESS
29 LDA /ENTRY              ;
30 STA AMPERV+2            ;
31 RTS
32 ;
33 ENTRY   JSR CHRGET       ;GET CHARACTER
34 JSR CHKOPN              ;SHOULD BE OPEN PAREN
35 JSR GETARYPT            ;GET ARRAY DESCRIPTOR
36 LDX #4                  ;
37 LDA (LOWTR),Y           ;SHOULD BE A ONE DIMENSION ARRAY
38 CMP #1                  ;
39 BEQ ENTRY1              ;
40 JMP SYNERR              ;ELSE DISPLAY ERROR MESSAGE
41 LDA LOWTR               ;SAVE ARRAY DESCRIPTOR ADRS.
42 STA SAVARRAY            ;LSB
43 LDA LOWTR+1             ;
44 STA SAVARRAY+1          ;MSB
45 JSR CHKCOM              ;CHK FOR COMMA + LOAD A W/NEXT BYTE
46 STA VARNAM              ;
47 JSR CHRGET              ;GET NEXT BYTE
48 BNE ENTRY2             ;SHOULD NOT BE END OF STATEMENT
49 JMP SYNERR              ;DISPLAY SYNTAX ERROR MESSAGE
50 CMP #'$'                ;DOLLAR SIGN
51 BNE NAMING              ;NO, MUST BE TWO CHARACTER NAME
52 LDA #000                ;
53 ORA #80                 ;NEGATIVE ASCII
54 STA VARNAM+1            ;
55 JSR FIND                ;FIND DESCRIPTOR
56 LDX #2                  ;
57 LDA (LOWTR),Y           ;GET + SAVE THE
58 STA VARLN               ;LENGTH OF PASSED STRING
59 INY                     ;
60 LDA (LOWTR),Y           ;GET + SAVE THE
61 STA VARAD               ;ADDRESS OF PASSED STRING
62 INY                     ;
63 LDA (LOWTR),Y           ;
64 STA VARAD+1             ;
65 LDA SAVARRAY            ;REESTABLISH LOWTR TO
66 STA LOWTR               ;ADDRESS OF ARRAY DESCRIPTOR
67 LDA SAVARRAY+1          ;
68 STA LOWTR+1             ;
69 LDX #5                  ;GET UPPER LIM. OF DIM (LOW BYTE)
70 LDA (LOWTR),Y           ;
71 STA UPLIM+1            ;MAKE LOW-HIGH
72 INY                     ;
73 LDA (LOWTR),Y           ;
74 STA UPLIM              ;
75 LDA #000                ;INITIALIZE LOWER LIMIT
76 STA LOWLIM              ;
77 STA LOWLIM+1            ;
78 JSR COMPLIM            ;INDEX=(UPLIM+LOWLIM)/2
79 JSR BY3                 ;MULTIPLY INDEX BY 3 (LENGTH OF PTR. ENTRIES)
80 CLC                    ;
81 LDA LOWTR               ;ADD BASE TO INDEX
82 ADC SAVARRAY            ;
83 STA LOWTR               ;TO OBTAIN POINTER TO ELEMENT
84 LDA LOWTR+1             ;
85 ADC SAVARRAY+1          ;
86 STA LOWTR+1             ;
87 LDX #7                  ;OFFSET TO LENGTH OF ELEMENT
88 LDA (LOWTR),Y           ;
89 STA ARRAYLN            ;
90 CMP VARLN               ;FIND SHORTEST ARGUMENT

```

```

9497 30 06      91      BML ARRAYST      ;ELEMENT SHORTEST
9499 AE 74 95    92      LDX VARLN      ;STRING SHORTEST
949C 4C A0 94    93      JMP CONT1
949F AA          94      ARRAYST TAX      ;PUT ELEMENT LENGTH IN X
94A0 C8          95      CONT1 INY      ;OFFSET TO ADDRESS
94A1 B1 9B      96      LDA (LOWTR),Y    ;GET LOW BYTE OF ADDRESS
94A3 8D 7F 95    97      STA ARRAYAD
94A6 C8          98      INY
94A7 B1 9B      99      LDA (LOWTR),Y    ;GET HIGH BYTE
94A9 8D 80 95    100     STA ARRAYAD+1
94AC A0 00      101     LDY #500      ;INITIALIZE Y
94AE AD 7F 95    102     LDA ARRAYAD    ;SET UP LOWTR AS
94B1 85 9B      103     STA LOWTR    ;ZERO PAGE PTR. FOR ARRAYAD
94B3 AD 80 95    104     LDA ARRAYAD+1
94B6 85 9C      105     STA LOWTR+1
94B9 B1 9B      106     COMPLP LDA (LOWTR),Y ;COMPARE ARRAY TO
94BA D1 83      107     CMP (VARAD),Y ;STRING
94BC 30 2F      108     BML STRNGHI ;STRING IS GREATER
94BE F0 03      109     BEQ COM1
94C0 4C 0F 95    110     JMP STRNGLO ;STRING IS LOWER
94C3 CB          111     COM1 INY
94C4 CA          112     DEX
94C5 D0 F1      113     BNE COMPLP ;CONTINUE COMPARE
94C7 AD 70 95    114     LDA ARRAYLN
94CA CD 74 95    115     CMP VARLN ;COMPARE STRING + ELEMENT LENGTH
94CD 30 1E      116     BML STRNGHI ;IF STRING IS LONGER
94CF F0 03      117     BEQ EXIT ;FOUND THE ELEMENT
94D1 4C 0F 95    118     JMP STRNGLO ;STRING IS SHORTER
94D4 A9 03      119     EXIT LDA #503 ;FIND OR CREATE A DESCRIPTOR
94D6 85 81      120     STA VARNAM ;FOR SS$ INTEGER
94D8 85 82      121     STA VARNAM+1
94DA 20 53 80    122     JSR FIND ;BY JSR TO FIND
94DD A0 02      123     LDY #2
94DF AD 76 95    124     LDA INDEX+1 ;STORE HIGH BYTE OF INDEX
94E2 91 9B      125     STA (LOWTR),Y ;FIRST
94E4 C8          126     INY
94E5 AD 75 95    127     LDA INDEX ;THEN LOW BYTE
94E8 91 9B      128     STA (LOWTR),Y
94EA 4C 95 D9    129     JMP DATA ;RESET TXTPTR + RETURN TO BASIC
94ED AD 79 95    130     STRNGHI LDA LOWLIM ;IF LOWLIM = INDEX
94F0 CD 75 95    131     CMP INDEX ;THAN ELEMENT CAN'T BE FOUND
94F3 D0 0B      132     BNE HI2
94F5 AD 7A 95    133     LDA LOWLIM+1
94F8 CD 76 95    134     CMP INDEX+1
94FB D0 03      135     BNE HI2
94FD 4C 4B 95    136     JMP NOTFOUND ;SO BRANCH TO NOTFOUND RTN.
9500 AD 75 95    137     HI2 LDA INDEX ;RESET LOWER LIMIT
9503 8D 79 95    138     STA LOWLIM
9506 AD 76 95    139     LDA INDEX+1
9509 8D 7A 95    140     STA LOWLIM+1
950C 4C 78 94    141     JMP SEARCHLPL ;CONTINUE SEARCH
950F AD 77 95    142     STRNGLO LDA UPLIM ;IF UPLIM=INDEX
9512 CD 75 95    143     CMP INDEX ;THEN ELEMENT CAN'T BE FOUND
9515 D0 0B      144     BNE LO2
9517 AD 78 95    145     LDA UPLIM+1
951A CD 76 95    146     CMP INDEX+1
951D D0 03      147     BNE LO2
951F 4C 4B 95    148     JMP NOTFOUND ;SO BRANCH TO NOTFOUND ROUTINE
9522 AD 75 95    149     LO2 LDA INDEX ;RESET UPPER LIMIT
9525 8D 77 95    150     STA UPLIM
9528 AD 76 95    151     LDA INDEX+1
952B 8D 78 95    152     STA UPLIM+1
952E 4C 78 94    153     JMP SEARCHLPL ;CONTINUE SEARCH
9531          154     ;
9531          155     ;COMPUTE NEW INDEX
9531 18          156     COMPIDX CLC ;INDEX=(UPLIM+LOWLIM)/2
9532 AD 77 95    157     LDA UPLIM ;ADD UPLIM TO LOWLIM
9535 6D 79 95    158     ADC LOWLIM
9538 8D 75 95    159     STA INDEX ;AND STOR IN INDEX
953B AD 78 95    160     LDA UPLIM+1
953E 6D 7A 95    161     ADC LOWLIM+1
9541 8D 76 95    162     STA INDEX+1
9544 4E 76 95    163     LSR INDEX+1 ;DIVIDE BY TWO
9547 6E 75 95    164     ROR INDEX
954A 60          165     RTS
954B A9 FF      166     NOTFOUND LDA #$FF ;-1 MEANS NOTFOUND
954D 8D 75 95    167     STA INDEX
9550 8D 76 95    168     STA INDEX+1
9553 4C D4 94    169     JMP EXIT
9556          170     ;
9556 AD 75 95    171     BY3 LDA INDEX ;LOWTR=(INDEX*3)
9559 85 9B      172     STA LOWTR
955B 06 9B      173     ASL LOWTR ;(LOWTR*2)
955D AD 76 95    174     LDA INDEX+1
9560 85 9C      175     STA LOWTR+1
9562 26 9C      176     ROL LOWTR+1
9564 18          177     CLC
9565 AD 75 95    178     LDA INDEX ;=CP STA LOWTR
956C AD 76 95    181     LDA INDEX+1
956F 65 9C      182     ADC LOWTR+1
9571 85 9C      183     STA LOWTR+1
9573 60          184     RTS
9574          185     ;
9574          186     ;INTERNAL STORAGE AREAS
9574          187     ;
9574          188     VARLN DFS $1 ;VARIABLES LENGTH
9575          189     INDEX DFS $2 ;SEARCH INDEX
9577          190     UPLIM DFS $2 ;HIGHEST POSSIBLE POSITION FOR SEARCH
9579          191     LOWLIM DFS $2 ;LOWEST POSSIBLE POSITION FOR SEARCH
957B          192     SAVARRAY DFS $2 ;WORK AREA
957D          193     ARRAYLN DFS $2 ;LENGTH OF CURRENT ARRAY ELEMENT
957F          194     ARRAYAD DFS $2 ;ADDRESS OF CURRENT ARRAY ELEMENT
9581          195     ;
9581          196     END

```

in SS%. To use the & feature you must BRUN the object program. The other choice is to BLOAD the program and use CALL -27632 in place of the ampersand. This will allow you to use this subroutine in conjunction with another ampersand routine.

Upon entering the subroutine at ENTRY the TXTPTR (see July Apple Slices for an explanation of TXTPTR, FIND, CHRGET, DATA, and VARNAM) is advanced to point at the first character past the GET token. Next, a JSR to CHKOPN (an Applesoft built-in routine) is performed, which checks for an open parenthesis. The JSR to GETARRAYPT (Applesoft built-in routine) returns with the address of the descriptor for XX\$ in LOWTR (9B\$ - 9C\$). If the array cannot be found an "OUT OF DATA IN LINE nnn" error message is produced.

Lines 36-40 check the number of dimensions to be sure that this is a one-dimensional array. If it is not, a syntax error message is produced (line 40). The array descriptor address is then saved for future use in SAVARRAY (lines 41 through 44). A JSR to CHKCOM ensures that a comma separates the two parameters and loads the accumulator with the first byte following the comma. This byte is stored at VARNAM. Lines 47 through 54 load VARNAM + 1 with either the negative ASCII of the second byte of the two-byte or longer variable name, or \$80 if the variable name is only one byte long.

A JSR to FIND loads LOWTR with the address of the descriptor of the passed variable. Lines 56 through 64 load and save the length and address of the passed variable in VARLN and VARAD respectively. Lines 65 through 74 re-establish LOWTR to the address of the array's descriptor (SAVARRAY) and initialize the upper limit (UPLIM) to the size of the array. The lower limit (LOWLIM) is then initialized to zero, and the main search loop (SEARCHLPL) is entered. First there is a JSR to COMPIDX, which is an internal routine that takes the average of the upper and lower limits and stores the result at INDEX. INDEX will be used as the current position in the array of the binary search.

Now SEARCHLPL takes the current value of the INDEX field and multiplies it by three (JSR BY3), placing the result in LOWTR. This is done because each string element in the array has a three-byte entry in the array descriptor, a

length byte followed by a two-byte address. To find the displacement of the individual element's entry from the base address of the array's descriptor, it is necessary to multiply INDEX by three.

LOWTR is then added to the base address of the array's descriptor (SAVARRAY); the result is stored back in LOWTR. The length of the searched element is then found and saved in ARRAYLN (lines 88 through 89). The seven-byte Y-index value is needed because the individual string array entries start seven bytes from the beginning of the array descriptor in any one-dimensional array. The X-register will be used as the number of bytes left in the array element and string variable to compare. It is initialized to the lower of the VARLN and ARRAYLN internal parameters (lines 90 through 94).

Next, the address of the array element is found and placed in LOWTR (lines 95 through 104). The compare loop (COMPLP) then compares the array element to the string variable, byte for byte, up to the length of the shortest of the two elements (using the

X-register as a counter). If the string is lower in value than the array element a JMP to STRNGLO is performed (line 110). If the string is higher in value, then a JMP to STRNGHI is performed (line 108). If the two items are equal (line 109) the lengths are compared. If the string is shorter it is considered to be lower in value and a JMP to STRNGLO is performed (line 116). If the two items are of equal length then a branch to EXIT is performed, which sets up an integer variable SS% and loads it with the current value of INDEX. This value is the location of the search argument in the array. The last thing EXIT does is JMP to DATA, which is Applesoft's routine to advance the TXTPTR to the end of the current statement (lines 119 through 129).

STRNGHI first compares the lower limit of the search (LOWLIM) to the INDEX. If they are equal then the upper limit and the lower limit have converged, which means the element could not be found. Under this condition a JMP to the internal routine NOTFOUND is performed (lines 130-136). NOTFOUND loads INDEX with a -1

and JMPs to EXIT where INDEX is passed to the SS% parameter as described above.

If the upper and lower limits have not converged, STRNGHI then resets the lower limit by moving INDEX (lines 137 through 140). STRNGHI then returns to the main search loop (SEARCHLP) to continue the search.

STRNGLO works essentially like STRNGHI except it tests for convergence by checking to see if INDEX is equal to the upper limit. If it is not, STRNGLO resets the upper limit to INDEX instead of the lower limit.

Subroutine Hints

Before using BINARY-SEARCH you should set HIMEM to 37888 or lower (if you decide to load the routine at \$9400). I could have set HIMEM for you in SETVEC, but I believe that leaving this task to you allows more flexibility; you can BLOAD and CALL the routine instead of using the & feature. You can also BRUN the subroutine from anywhere in your BASIC program, instead of just from the first line.

MICRO™

ANNOUNCING A NEW JOURNAL

DEVOTED TO ALL ASPECTS

OF MICROCOMPUTER

USE AT THE

UNDERGRADUATE

LEVEL

PREMIER

ISSUE

FEBRUARY 1983

write:

Collegiate Microcomputer

Rose-Hulman Institute of Technology

Terre Haute IN 47803 USA

Prospectus sent upon request.

Decision Systems

Decision Systems
P.O. Box 13006
Denton, TX 76203

SOFTWARE FOR THE APPLE II™

ISAM-DS is an integrated set of Applesoft routines that gives indexed file capabilities to your BASIC programs. Retrieve by key, partial key or sequentially. Space from deleted records is automatically reused. Capabilities and performance that match products costing twice as much.
\$50 Disk, Applesoft.

PBASIC-DS is a sophisticated preprocessor for structured BASIC. Use advanced logic constructs such as IF...ELSE..., CASE, SELECT, and many more. Develop programs for Integer or Applesoft. Enjoy the power of structured logic at a fraction of the cost of PASCAL.
\$35 Disk, Applesoft (48K, ROM or Language Card)

DSA-DS is a dis-assembler for 6502 code. Now you can easily dis-assemble any machine language program for the Apple and use the dis-assembled code directly as input to your assembler. Dis-assembles instructions and data. Produces code compatible with the S-C Assembler (version 4.0), Apple's Toolkit assembler and others.
\$25 Disk, Applesoft (32K, ROM or Language Card)

FORM-DS is a complete system for the definition of input and output forms. FORM-DS supplies the automatic checking of numeric input for acceptable range of values, automatic formatting of numeric output, and many more features.
\$25 Disk, Applesoft (32K, ROM or Language Card)

UTIL-DS is a set of routines for use with Applesoft to format numeric output, selectively clear variables (Applesoft's CLEAR gets everything), improve error handling, and interface machine language with Applesoft programs. Includes a special load routine for placing machine language routines underneath Applesoft programs.
\$25 Disk, Applesoft

SPEED-DS is a routine to modify the statement linkage in an Applesoft program to speed its execution. Improvements of 5-20% are common. As a bonus, SPEED-DS includes machine language routines to speed string handling and reduce the need for garbage clean-up. Author: Lee Meador.
\$15 Disk, Applesoft (32K, ROM or Language Card)

(Add \$4.00 for Foreign Mail)

*Apple II is a registered trademark of the Apple Computer Co.



BOX 120
ALLAMUCHY, N.J. 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

THE TASK* MASTERS

HDE supports the *TIM, AIM, SYM and KIM (TASK) with a growing line of computer programs and peripheral components. All HDE component boards are state-of-the-art 4½" x 6½", with on board regulation of all required voltages, fully compatible with the KIM-4 bus.

OMNIDISK 65/8 and 65/5

Single and dual drive 8" and 5¼" disk systems. Complete, ready to plug in, bootstrap and run. Include HDE's proprietary operating system, FODS (File Oriented Disk System).

DM816-M8A

An 8K static RAM board tested for a minimum of 100 hours and warranted for a full 6 months.

DM816-UB1

A prototyping card with on-board 5V regulator and address selection. You add the application.

DM816-P8

A 4/8K EPROM card for 2708 or 2716 circuits. On board regulation of all required voltages. Supplied without EPROMS.

DM816-CC15

A 15 position motherboard mounted in a 19" RETMA standard card cage, with power supply. KIM, AIM and SYM versions.

DISK PROGRAM LIBRARY

Offers exchange of user contributed routines and programs for HDE Disk Systems. Contact Progressive Computer Software, Inc. for details.

HDE DISK BASIC

A full range disk BASIC for KIM based systems. Includes PRINT USING, IF . . . THEN . . . ELSE. Sequential and random file access and much more. \$175.00

HDE ADVANCED INTERACTIVE DISASSEMBLER (AID)

Two pass disassembler assigns labels and constructs source files for any object program. Saves multiple files to disk. TIM, AIM, SYM, KIM versions. \$95.00

HDE ASSEMBLER

Advanced, two pass assembler with standard mnemonics. KIM, TIM, SYM and KIM cassette versions. \$75.00 (\$80.00 cassette)

HDE TEXT OUTPUT PROCESSING SYSTEM (TOPS)

A comprehensive text processor with over 30 commands to format and output letters, documents, manuscripts. KIM, TIM and KIM cassette versions. \$135.00 (\$142.50 cassette)

HDE DYNAMIC DEBUGGING TOOL (DDT)

Built in assembler/disassembler with program controlled single step and dynamic breakpoint entry/deletion. TIM, AIM, SYM, KIM AND KIM cassette versions. \$65.00 (\$68.50 cassette)

HDE COMPREHENSIVE MEMORY TEST (CMT)

Eight separate diagnostic routines for both static and dynamic memory. TIM, AIM, SYM, KIM and KIM cassette versions. \$65.00 (\$68.50 cassette)

AVAILABLE DIRECT OR FROM THESE FINE DEALERS:

Progressive Computer Software
405 Corbin Road
York, PA 17403
(717) 845-4954

Johnson computers
Box 523
Medina, Ohio 44256
(216) 725-4560

Lux Associates
20 Sunland Drive
Chico, CA 95926
(916) 343-5033

Falk-Baker Associates
382 Franklin Avenue
Nutley, NJ 07110
(201) 661-2430

Laboratory Microcomputer Consultants
P.O. Box 84
East Amherst, NY 14051
(716) 689-7344

Perry Peripherals
P.O. Box 924
Miller Place, NY 11764
(516) 744-6462

Adding Voice to a Computer

by Michael E. Valdez

A low-cost procedure for sampling and reproducing voice with a computer including the required hardware and software.

Voice

requires:

A computer with a 4-bit port available and a Motorola 3417 speech/digital converter

Several methods are available today to add voice to a computer. The method developed by Texas Instruments uses a model of the mouth and generates the necessary parameters by linear predictive coding. This method gives excellent results producing isolated words with very high quality, but is expensive. Another problem is that it is necessary to have a read-only memory with the parameters of the words to be used; this read-only memory can be produced only by Texas Instruments. It has several ready-made, read-only memories with standard vocabularies at a very reasonable price. Using this method requires minimal knowledge of acoustics and linguistics. The user has to write some simple programs to control the unit, the worst requirement being to prevent the words from running together.

The signal compression and delta modulation method developed by National Semiconductors, although very different technically, is similar from the user's point of view to the one developed by Texas Instruments. With this method it is also necessary to use a read-only memory produced by the manufacturer, and the cost is also in the same range (around two hundred dollars). But, the results are somewhat robotic.

A continuously variable slope delta modulation developed by Motorola uses the same integrated circuit for storing and reproducing speech. This is

the only method available today that permits the user to sample his own speech. The unit to be described in this article is inexpensive (fifteen dollars for parts), and the knowledge requirements of acoustics and linguistics are minimal. The user should know how to use a tape recorder and write some simple programs. The hardest requirement is the timing of the loops. The quality of reproduction is quite good and depends heavily on the quality of the tape recording equipment. The digital data can be stored in read-write or read-only memory, or it can be saved on magnetic tape or disk.

The phoneme concatenation method uses the SC01 phoneme synthesizer developed by Votrax. The results of this procedure are mechani-

cal but it is important to recognize that this is the only real synthesis procedure for the production of speech by a computer; that is, it is not necessary to sample speech to obtain data to be reproduced by the computer as in the other methods. The voice is generated by entering numbers into the computer and the SC01, or any other device. Naturally, since this method does not reproduce speech, the generated voice does not resemble the voice of the operator, or anybody else. In its most elementary use, the voice can be described as robot-like because of the lack of intonation and inflections. With additional work and knowledge, it is possible to obtain better results. The cost of a simple unit is under one hundred dollars. The use of this method re-

Listing 1: Program for Adding Voice to a Computer

```
1000:      2          ORG $1000
1000:      3 * MODIFY TO SUIT INSTALLATION
1000:      4 *****
1000:      5 *
1000:      6 *
1000:      7 * PROGRAM TO ADD VOICE TO ANY *
1000:      8 *
1000:      9 *
1000:     10 *          COMPUTER
1000:     11 *
1000:     12 *
1000:     13 *****
1000:     14          MSB OFF
1000:     15 *
1000:     16 * STORAGE LOCATION MUST BE MODIFIED
1000:     17 * TO SUIT SYSTEM
1000:     18 *
0010:     19 FNT      EQU $10
0012:     20 END      EQU $12
0014:     21 BITS     EQU $14
1000:     22 *
1000:     23 * SYSTEM SUBROUTINES
1000:     24 *
F8B2:     25 KKK      EQU $F8B2    KEYBOARD INPUT IN ASCII
FA5F:     26 OUT      EQU $FA5F    OUTPUT IN ASCII
1000:     27 *
1000:     28 * LOCATIONS OF I/O PORT
1000:     29 *
EF80:     30 DELR     EQU $EF80    6522 PORT
EF82:     31 DELDR    EQU $EF82    6522 DATA DIRECTION REGISTER
1000:     32 *
1000:     33 *
1000:     34 * PROGRAM START
1000:     35 *
1000:     36 *
1000:A2 00     37 DELTA   LDX #0          BEGINNING OF BUFFER
1002:ED 52 11   38 DEL1    LDA DLM,X
1005:C9 1F     39          CMP #$1F
1007:F0 06     40          BEQ DEL4
```

(continued)

quires some knowledge of linguistics and phonetics if good results are desired, but the manufacturer provides substantial support.

Intel has developed what they call an analog microprocessor — a single-chip device to work with analog signals. This unit, the 2920, can be used for speech synthesis or reproduction, but its use is limited to those persons with a substantial knowledge of acoustics, linguistics, physics, mathematics, and a high level of programming proficiency. This unit is for the serious user. There are several other units in this category, manufactured by TRW, Harris, and others.

The Motorola 3417

The Motorola 3417 is a linear bipolar chip housed in a 16-pin dual in line package, which is compatible with both TTL and CMOS technologies. The 16-pin package makes it easy to mount since sockets are available everywhere. The chip has the circuitry for the encoder (speech to digital) and decoder (digital to speech) conversions.

Pins 1 and 7 are the speech input and output while pins 13 and 9 are the digital input and output, respectively. Data then travels in the chip from pin 1 to pin 9 or from pin 13 to pin 7 depending on the input to pin 15, encode/decode. A high in pin 15 makes the chip encode the speech input to pin 1 giving a digital output through pin 9. A low in pin 15 converts digital input through pin 13 to a speech output in pin 7.

The chip provides for positive and negative excursion of the speech signal with a regulated voltage at half of the supply voltage that is used as zero for the speech input or output. The chip also provides pin 12 to set the threshold between digital zero and one, to adjust the chip to different technologies. The feedback point of the output amplifier is accessible in pin 6 to include a filter if desired. Pins 3, 4, and 11 provide access to the integrator to permit the addition of a syllabic filter. The Motorola 3417 works with a single supply voltage and requires a 16 KHz clock input at pin 14.

The data sheet provides a full explanation of the theory of continuously variable delta modulation as well as a variety of circuit information.

Hardware

For reasons of simplicity and low cost, the unit described in this article

Listing 1 (continued)

1009:20 5F FA	41	JSR	OUT	
100C:E8	42	INX		
100D:D0 F3	43	BNE	DEL1	
100F:A9 0E	44	LDA	\$\$E	INITIALIZE PORT
1011:8D 82 EF	45	STA	DELR	
1014:20 02 11	46	JSR	ADRS	
1017:D0 01	47	BNE	DEL2	
1019:	48	* PROGRAM ENDS WHEN THE INITIAL ADDRESS IS ZERO		
1019:60	49	RTS		
101A:C9 FF	50	DEL2	CMP	\$\$FF STANDARD FILE
101C:F0 1A	51	BEQ	DEL3	
101E:A5 13	52	LDA	END+1	MOVE TO POINTER
1020:85 11	53	STA	PNT+1	
1022:A5 12	54	LDA	END	
1024:85 10	55	STA	PNT	
1026:A2 00	56	LDX	#0	END OF BUFFER
1028:ED FA 11	57	DEL5	LDA	DLM3,X
102E:C9 1F	58	CMP	\$\$1F	
102D:F0 06	59	BEQ	DEL6	
102F:20 5F FA	60	JSR	OUT	
1032:E8	61	INX		
1033:D0 F3	62	BNE	DEL5	
1035:20 02 11	63	DEL6	JSR	ADRS
1038:A2 00	64	DEL3	LDX	#0 INPUT OR OUTPUT?
103A:8D E1 11	65	DEL7	LDA	DLM1,X
103D:C9 1F	66	CMP	\$\$1F	
103F:F0 06	67	BEQ	DEL8	
1041:20 5F FA	68	JSR	OUT	
1044:E8	69	INX		
1045:D0 F3	70	BNE	DEL7	
1047:20 82 F8	71	DEL8	JSR	KKK
104A:C9 4F	72	CMP	\$\$4F	ASCII 0
104C:F0 58	73	BEQ	OUTPUT	
104E:C9 49	74	CMP	\$\$49	ASCII I
1050:D0 E6	75	BNE	DEL3	
1052:	76	*		
1052:	77	*		
1052:	78	* INPUT ROUTINE		
1052:	79	*		
1052:	80	*		
1052:A2 00	81	LDX	#0	SIGNAL WHEN READY
1054:8D 16 12	82	INP0	LDA	DLM2,X
1057:C9 1F	83	CMP	\$\$1F	
1059:F0 06	84	BEQ	INP4	
105B:20 5F FA	85	JSR	OUT	
105E:E8	86	INX		
105F:D0 F3	87	BNE	INP0	
1061:20 82 F8	88	INP4	JSR	KKK
1064:A9 0C	89	LDA	\$\$C	START CLOCK
1066:8D 80 EF	90	STA	DELR	
1069:A0 00	91	LDY	#0	
106B:A2 08	92	INP4	LDX	#8 EIGHT BITS
106D:A9 04	93	INP1	LDA	#4 CLOCK LOW
106F:8D 80 EF	94	STA	DELR	
1072:EA	95	NOP		DUMMY
1073:EA	96	NOP		DUMMY
1074:AD 80 EF	97	LDA	DELR	GET NEXT BIT
1077:4A	98	LSR	A	MOVE TO CARRY FLAG
1078:26 14	99	ROL	BITS	ASSEMBLE WORD
107A:A9 0C	100	LDA	\$\$C	CLOCK HIGH
107C:8D 80 EF	101	STA	DELR	
107F:CA	102	DEX		COUNT BITS
1080:D0 18	103	BNE	INP3	CYCLE EIGHT TIMES
1082:A5 14	104	LDA	BITS	RECOVER WORD
1084:91 10	105	STA	(PNT),Y	SAVE IN BUFFER
1086:E6 10	106	INC	PNT	INCREMENT POINTER
1088:D0 02	107	BNE	INP2	
108A:E6 11	108	INC	PNT+1	
108C:38	109	INP2	SEC	TEST FOR BUFFER FULL
108D:A5 12	110	LDA	END	
108F:E5 10	111	SBC	PNT	
1091:A5 13	112	LDA	END+1	
1093:E5 11	113	SBC	PNT+1	
1095:E0 D4	114	BCC	INPUT	GO BACK FOR MORE
1097:4C 00 10	115	JMP	DELTA	END
109A:A1 14	116	INP3	LDA	(BITS,X) DUMMY
109C:A1 14	117	LDA	(BITS,X)	DUMMY
109E:A1 14	118	LDA	(BITS,X)	DUMMY
10A0:A1 14	119	LDA	(BITS,X)	DUMMY
10A2:85 14	120	LDA	BITS,X	DUMMY
10A4:85 14	121	LDA	BITS,X	DUMMY
10A6:4C 6D 10	122	JMP	INP1	CONTINUE
10A9:	123	*		
10A9:	124	*		
10A9:	125	* OUTPUT ROUTINE		
10A9:	126	*		
10A9:	127	*		
10A9:A9 00	128	OUTPUT	LDA	#0 CLOCK LOW
10AB:8D 80 EF	129	STA	DELR	

Listing 1 (continued)

```

10AE:A2 00 130 LDX #0 SIGNAL WHEN READY
10E0:ED 16 12 131 OUT4 LDA DLM2,X
10E3:C9 1F 132 CMP ##1F
10E5:F0 06 133 BEQ OUT5
10E7:20 5F FA 134 JSR OUT
10EA:E8 135 INX
10EE:D0 F3 136 BNE OUT4
10E0:20 E2 F8 137 OUT5 JSR KKK
10C0:A0 00 138 LDY #0
10C2:E1 10 139 OUT0 LDA (PNT),Y GET NEXT WORD
10C4:85 14 140 STA BITS SAVE IT IN BITS
10C6:E6 10 141 INC PNT INCREMENT POINTER
10C8:D0 02 142 BNE OUT1
10CA:E6 11 143 INC PNT+1
10CC:A2 08 144 OUT1 LDX #8 SEND EIGHT BITS
10CE:A9 08 145 OUT2 LDA #8 CLOCK HIGH
10D0:80 80 EF 146 STA DELR
10D3:A9 02 147 LDA #2 PREPARE ACCUMULATOR
10D5:06 14 148 ASL BITS GET BIT
10D7:2A 149 ROL A INTO ACCUMULATOR
10D8:2A 150 ROL A SHIFT ONE MORE
10D9:80 80 EF 151 STA DELR SEND TO 3417
10DC:29 02 152 AND #2 CLEAR CLOCK
10DE:80 80 EF 153 STA DELR CLOCK LOW
10E1:CA 154 DEX EIGHT BITS?
10E2:D0 0E 155 ENE OUT3 GO FOR MORE
10E4:38 156 SEC TEST FOR BUFFER FULL
10E5:A5 12 157 LDA END
10E7:E5 10 158 SBC PNT
10E9:A5 13 159 LDA END+1
10EB:E5 11 160 SBC PNT+1
10ED:E0 D3 161 BCS OUT0 GO FOR MORE
10EF:4C 00 10 162 JMP DELTA
10F2:A1 14 163 OUT3 LDA (BITS,X) DUMMY
10F4:A1 14 164 LDA (BITS,X) DUMMY
10F6:A1 14 165 LDA (BITS,X) DUMMY
10F8:E1 14 166 LDA (BITS),Y DUMMY
10FA:E5 14 167 LDA BITS,X DUMMY
10FC:E5 14 168 LDA BITS,X DUMMY
10FE:EA 169 NOP DUMMY
10FF:4C CE 10 170 JMP OUT2 CONTINUE
1102: 171 *
1102: 172 *
1102: 173 * GET ADDRESS SUBROUTINE
1102: 174 *
1102: 175 *
1102:A9 00 176 ADRS LDA #0
1104:85 12 177 STA END
1106:85 13 178 STA END+1
1108:20 E2 F8 179 ADRO JSR KKK GET CHARACTER
110E:20 5F FA 180 JSR OUT DISPLAY IT
110E:C9 53 181 CMP ##53 CHECK IF S
1110:D0 11 182 BNE ADR1
1112:A9 00 183 LDA #0 STANDARD BUFFER
1114:85 10 184 STA PNT
1116:84 12 185 STY END CHANGE VALUES
1118:A9 04 186 LDA #4
111A:85 11 187 STA PNT+1 PER INSTALLATION
111C:A9 40 188 LDA ##40
111E:85 13 189 STA END+1
1120:A9 FF 190 LDA ##FF
1122:60 191 RTS
1123:C9 0D 192 ADR1 CMP ##D CHECK FOR CAR RET
1125:F0 26 193 BEQ ADR3
1127:C9 30 194 CMP ##30 TEST IF NUMBER
1129:90 0D 195 BCC ADR0 IGNORE IF NOT
112E:C9 3A 196 CMP ##3A
112D:90 0C 197 BCC PKA
112F:C9 41 198 CMP ##41 TEST IF HEXA LETTER
1131:90 D5 199 BCC ADR0 IGNORE IF NOT
1133:29 5F 200 AND ##5F CONVERT TO UPPER CASE
1135:C9 47 201 CMP ##47
1137:E0 CF 202 BCS ADR0
1139:69 09 203 ADC #9
113E:29 0F 204 PKA AND ##F
113D:0A 205 ASL A ROL INTO END, END+1
113E:0A 206 ASL A
113F:0A 207 ASL A
1140:0A 208 ASL A
1141:A2 04 209 LDX #4
1143:0A 210 ADR2 ASL A
1144:26 12 211 ROL END
1146:26 13 212 ROL END+1
1148:CA 213 DEX
1149:D0 F8 214 BNE ADR2
114B:F0 EB 215 BEQ ADR0
114D:A5 12 216 ADR3 LDA END GET IF ZERO
114F:05 13 217 ORA END+1
1151:60 218 RTS

```

(continued)

uses the Motorola MC3417 continuously variable delta modulator/demodulator. The Harris HC55516 could also be used but the circuit must be redesigned to account for the fact that the 55516 is a CMOS chip. If the computer to be used has an available port with four free bits, very few additional components are needed. Furthermore, none of the components shown on the circuit is critical and the values can vary before the quality of the results is degraded. Normally, the noise and the quality of the tape recording equipment will be the limiting factors for the quality of the reproduction. The circuit shows part of a 6522 Versatile Interface Adapter controlling the 3417, but the job can be done with any other programmable parallel port, or with three flip-flops and one tri-state unit. If the program presented with this article is to be used, the location of each signal in the word must be respected. Bit zero is the digital output from the chip, bit one is the digital input to the chip, bit two is the encode/decode control, and bit three is the clock. Bit zero must be programmed as input and the other three as outputs.

One interesting point to mention in this circuit is the lack of a clock. The 3417 requires a 16 KHz clock; in this circuit the clock is produced in software thereby avoiding the problems of synchronization. If an independent clock is used, it is necessary to sample it to send and recover the bits at the proper time.

The audio amplifier shown on the circuit is very simple and includes an elementary filter to reduce the digitizing noise. Notice the capacitor in parallel with the speaker for the same reason. Some experimentation with the values used in a particular circuit might improve the quality of reproduction. The circuit can be built in the existing board of the computer, if there is room, or wire wrapped in a small board and connected as convenient. Only five volts are required to power the unit.

Software

The software presented with this article is self explanatory. The user must adjust the memory locations to match his system. The subroutine KKK reads the keyboard and returns with the ASCII character in the accumulator; the subroutine OUT displays the accumulator.

The only part of the program that

should be treated carefully is the generation of the clock. It is important to maintain the sampling and reproduction clocks as close as possible. Large variations produce unpleasant results.

The program presented here has been written for the 6502. Converting the code to any other microprocessor requires only limited programming ability.

The Use of the Unit

The unit is very simple to use. A cassette or any tape recorder records the words of messages to be stored for later reproduction. It is good to leave pauses before and after each part to aid in recognition. When an acceptable record has been obtained, especially without too much background noise, the output of the tape recorder is connected to the input of the unit, and the program is run.

Some practice is required to start the tape recorder and to signal the computer such that the whole record is sampled; this is especially true when the record is long and the buffer is small. Recall that 2K of memory is needed for each second of speech. The program permits finding the initial and final location of memory used by the

Listing 1 (continued)

```

1152:          219 *
1152:53 50 45 220 DLM      ASC  "SPEECH  ANALYSIS AND SYNTHESIS USING"
1155:45 43 48
1158:20 41 4E
1158:41 4C 59
115E:53 49 53
1161:20 41 4E
1164:44 20 53
1167:59 4E 54
116A:48 45 53
116D:49 53 20
1170:55 53 49
1173:4E 47
1175:00          221      DFB  13
1176:43 4F 4E 222      ASC  "CONTINUOUSLY VARIABLE SLOPE DELTA
                        MODULATION"

1179:54 49 4E
117C:55 4F 55
117F:53 4C 59
1182:20 56 41
1185:52 49 41
1188:42 4C 45
118E:20 53 4C
118E:4F 50 45
1191:20 44 45
1194:4C 54 41
1197:20 4D 4F
119A:44 55 4C
119D:41 54 49
11A0:4F 4E
11A2:00          223      DFB  13
11A3:57 49 54 224      ASC  "WITH  THE MOTOROLA MC3417 IC."

11A6:48 20 54
11A9:48 45 20
11AC:4D 4F 54
11AF:4F 52 4F
11B2:4C 41 20
11B5:4D 43 33
11B8:34 31 37
11BB:20 49 43
11BE:2E
11BF:0D 0D          225      DFB  13,13
11C1:50 4C 45 226 DLM0  ASC  "PLEASE,  ENTER BEGINING ADDRESS"

11C4:41 53 45
11C7:2C 20 45
11CA:4E 54 45
11CD:52 20 42
11D0:45 47 49
11D3:4E 49 4E
11D6:47 20 41
11D9:44 44 52
11DC:45 53 53
11DF:0D 1F          227      DFB  13,$1F
11E1:          228 *
11E1:0D          229 DLM1  DFB  13
11E2:49 53 20 230      ASC  "IS  IT INPUT OR OUTPUT?"

11E5:49 54 20
11E8:49 4E 50
11EB:55 54 20
11EE:4F 52 20
11F1:4F 55 54
11F4:50 55 54
11F7:3F
11F8:0D 1F          231      DFB  13,$1F
11FA:          232 *
11FA:50 4C 45 233 DLM3  ASC  "PLEASE,  ENTER LAST ADDRESS"

11FD:41 53 45
1200:2C 20 45
1203:4E 54 45
1206:52 20 4C
1209:41 53 54
120C:20 41 44
120F:44 52 45
1212:53 53
1214:0D 1F          234      DFB  13,$1F
1216:          235 *
1216:50 4C 45 236 DLM2  ASC  "PLEASE,  SIGNAL WHEN READY"

1219:41 53 45
121C:2C 20 53
121F:49 47 4E
1222:41 4C 20
1225:57 48 45
1228:4E 20 52
122B:45 41 44
122E:59
122F:0D 1F          237      DFB  13,$1F
1231:          238 *
*** SUCCESSFUL ASSEMBLY: NO ERRORS

```

Scotch®

MEMOREX

Verbatim.

maxell.

BASF

wabash

Diskettes and all your media needs
Our **REGULAR** prices are **SPECIAL**

CALL FREE (800) 421-3957

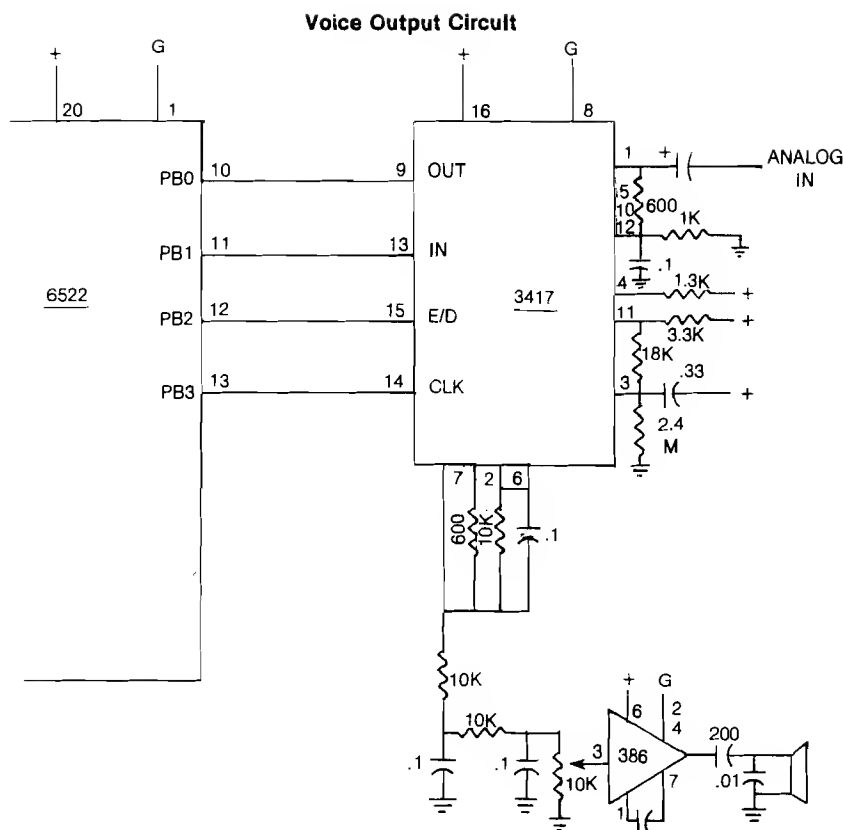
C.O.D. charge cards accepted.
Excellent dealer program.

Softel

1418 West Shaw Avenue
Fresno, CA 93711

In Cal. call **(209) 221-1118**

Foothill of The Sierras



sample, by changing the initial and final locations of the part to be reproduced.

If the message has pauses, it is possible to save memory by converting the reproduction program into a subroutine, making a call for each one of the parts, with appropriate waiting loops separating them. If it is better to leave the pauses in, clear the tape noise by storing hexadecimal 55 in all the locations of the pause. Now it is possible to see how little noise the process itself introduces!

When the message is to be stored in permanent memory and used many times, it is advisable to use a good high-speed tape recorder and a person with a pleasant voice to produce the originals. With several messages stored on disk it is possible to write a routine that calls the proper message into a standard area of memory and reproduces it. In this way, the same routine can handle many messages in an economical way.

You may contact Mr. Valdez at 1001 Flotilla, Indian Harbour Beach, FL 32937.

MICRO

PERRY PERIPHERALS REPAIRS KIMs!! (SYM_s AND AIM_s TOO)

- We will Diagnose, Repair, and Completely Test your Single Board Computer
- We Socket all replaced Integrated Circuits
- You receive a 30-day Parts and Labor Warranty
- Your repaired S.B.C. returned via U.P.S. — C.O.D., Cash

Don't delay! Send us your S.B.C. for repair today
Ship To: (Preferably via U.P.S.)

PERRY PERIPHERALS

6 Brookhaven Drive
Rocky Point, NY 11778

KIM-1 REPLACEMENT MODULES

- Exact replacement for MOS/Commodore KIM-1 S.B.C.
- Original KIM-1 firmware — 1K and 4K RAM versions

REPLACEMENT KIM-1 KEYBOARDS

- Identical to those on early KIMS — SST switch in top right corner
- Easily installed in later model KIMS

Perry Peripherals is an authorized HDE factory service center.

Perry Peripherals carries a full line of the acclaimed HDE expansion components for you KIM, SYM, and AIM, including RAM boards, Disk Systems, and Software like HDE Disk BASIC V1.1. Yes, we also have diskettes. For more information write to: P.O. Box 924, Miller Place, NY 11764, or Phone (516) 744-6462.

Enhanced Video for OSI C1P

by David Cantrell and Terry Terrance

Add a screen blanker, inverse upper case, and dim character set to your Challenger.

Enhanced Video requires:

OSI C1P
hardware modification

By adding five chips and cutting only two traces, you can add several features to your C1P video section. There will be a trade-off for these features, however. To keep the hardware and software as simple as possible, you lose lower-case alphanumerics when these features are implemented. But, no software support is necessary; no cumbersome POKEing and no software drivers to scroll a background screen (because there isn't any). You simply release your SHIFT-LOCK key whenever you want to enter modified video. Your machine's video will interpret lower-case characters as modified video whenever this modification is enabled. Since the rest of your machine simply "sees" lower-case alphanumerics, they can be put into strings and then simply PRINTed to the screen. The video modification can be disabled with either a hardware or software switch.

The circuit keys on Video Data Bit 5 (VD5) and Video Data Bit 6 (VD6). Whenever these bits are high and the modification is enabled, VD5 and VD6 will be masked, turning lower case into upper case, and an upper-case character in the selected "mode" (i.e., inverse, dim, etc.) will be displayed instead of the lower-case character. Since characters above 128 also have VD5 and/or VD6 set, gating is used to restore VD5 and VD6 and disable the modification whenever VD7 is set, retaining your graphics characters.

Before we get into soldering, let's

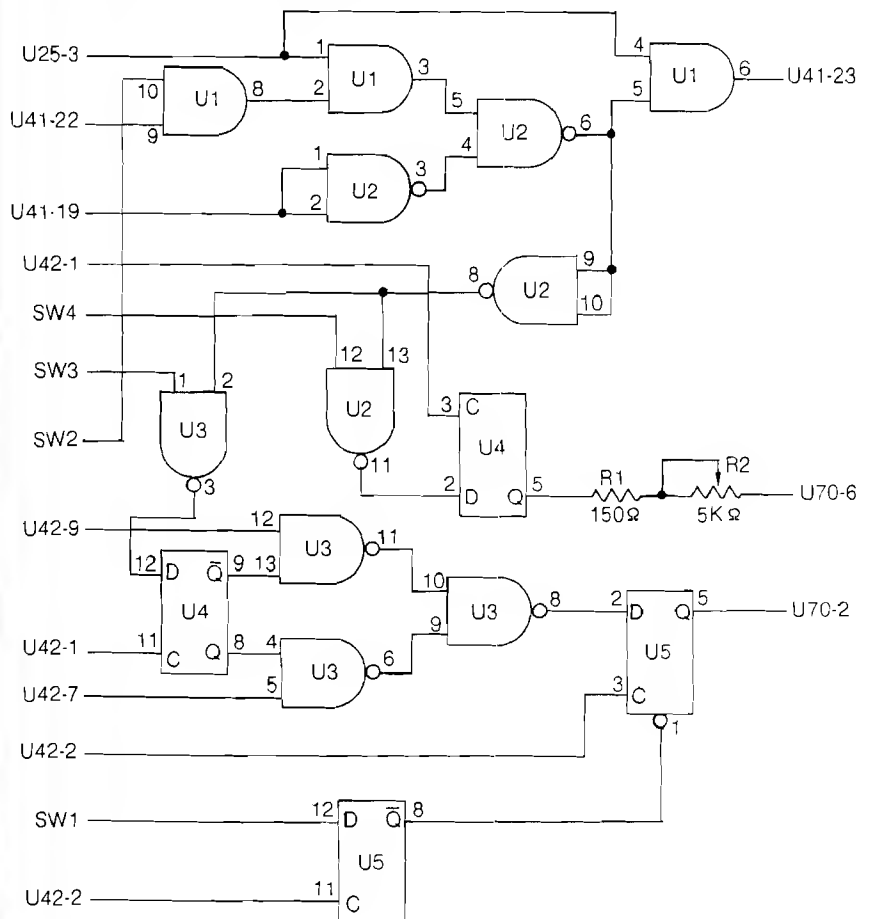
discuss OSI's video as implemented on the C1P. Even though we've spent the past couple of years squinting at our C1P's screen almost daily, some of its subtleties have escaped us. When the screen is filled with CHR\$(161) (OSI's solid white block character) and is viewed from about two feet away, all but the poorest TV or video monitor will show faint dark vertical lines on character cell boundaries. You may have attributed these lines to a one-dot-wide intercell space.

Closer inspection reveals that the whole screen is filled with evenly

spaced dots — no blank spaces appear between cells. As the rows of dots of each character are clocked out of the shift register U42, the first dot in each row is held only one-third as long as the others in that row. Since this happens for the first dot of each row and for each character, the end result is faint dark bars when viewed from a distance.

This is the subtle video defect alluded to before. It's so subtle that most OSIs do not notice it, or pass it off as intercell spacing. If C4 users are wondering why this effect can't be seen, the effect is reversed on the C4. The first

Figure 1: Schematic for Enhanced Video



dot is accentuated giving rise to bright vertical lines. This minor problem wouldn't be worth mentioning except the timing defect that causes it must be fixed if we are to add our modified video.

Before you begin construction, here are a few warnings. Keep all wires as short and as direct as possible. You'll be dealing with your video signal at RF frequencies. You'll want to avoid re-radiating your game of invaders all over your house and quite possibly to the neighbors' too. Do not substitute 74LSXX series components for 74XX series components or *vice versa*. This circuit is carefully balanced regarding timing and current drive capabilities; tampering will probably overheat all of the components in the circuit.

The parts list is short; you will need

U1	74LS08	Quad 2-Input And Gates
U2, U3	74LS00	Quad 2-Input Nand Gates
U4, U5	7474	Dual D Flip-Flop
R1		150 Ohm resistor
R2		5K Ohm potentiometer
SW1-SW4		SPST switch

Since there are five chips in the circuit, it cannot be assembled in the proto area of your C1P. You can assemble the circuit on perfboard or solderless breadboard using wire-wrap (or any technique you prefer). The circuit assembles in a straightforward manner. In figure 1 the chips numbered U1-U5 refer to the components of our modification; all other "U" numbers refer to chips on your C1P.

The schematic does not show how to wire in SW1-SW4. SW1-SW4 are the mode selection switches; each one should connect its associated line to ground. We have not found it necessary, but good circuit design would dictate that the lines SW1-SW4 should be pulled up to +5 by 3.3K pull-up resistors. Figure 1 does not show supplying +5V and ground to all of the chips in the circuit. All the chips used have the standard DIP power and ground pins. For 14-pin packages, all pins 7 should be wired to ground and all pins 14 should be supplied with +5V.

Once the circuit is assembled, you must splice it onto your C1P. Cut the trace running from U41 pin 23 to U40 pin 13, and the trace running from U42 pin 9 to U70 pin 2. Connect U25 pin 3 to U1 pin 1. Connect U41 pin 22 to U1 pin 9 and U41 pin 19 to U2 pin 2. Connect U1 pin 6 to U41 pin 23.

We'll stop for a moment and explain what this part of the circuit does. U25 pin 3 is VD5 and U41 pin 22 is VD6, the data bits that the circuit keys on to know whether to output modified video. U41 pin 19 is VD7. Three gates of U1 and two gates of U2 perform logic to accomplish the following functions. If VD5 and VD6 are high and SW2 is high and VD7 is low, U1 pin 6 is low causing lower-case characters to be read as upper case and activating the rest of the circuit via U2 pins 9 and 10. If either VD6 or VD5 is low or SW2 is low, U1 pin 6 will be high and the screen will behave normally.

Continuing with connections, U42 pin 9 is brought into U3 pin 12. U42 pin 1 is brought into U4 pin 11; U42 pin 7 is brought into U3 pin 5. Connect U42 pin 2 to U5 pin 3 and connect U42 pin 2 to U5 pin 8. Signals coming out of the circuit on U5 pin 5 must be connected to U70 pin 2. The output of the potentiometer R2 should be brought to U70 pin 6.

This is where our circuit starts modifying video. If the first part of the circuit has recognized a modified video situation (i.e., VD5 VD6 VD7 SW2), then U2 pin 8 goes high. The signal is now fed to parts of U2 and U3 where, combined with the states of switches SW3 and SW4, the inverse and dim options are selected. If dim is selected, either alone or in combination with inverse, the signal on U2 pin 11 is used to enable the flip-flop U4, which is clocked at the shift-load rate (i.e., CLK/8) and through the R1-R2 network modulates the video for a dimming effect. R2 controls the level of brightness from almost fully bright to almost dark. SW3 controls the inverse option. If it is low, the normal video signal is passed from U42 pin 9 out to U5 pin 5 without inversion (but with latching as we will see in a moment). When SW3 is high, the shift-load clock (from U42 pin 1) and the inverse shift register output are combined by sections of U4 and U3 to produce inverse video. The section of U5 that immediately follows fixes the video defect we mentioned earlier. Instead of the dots being cut off by the video chain clock, it is now latched for the whole period of the system clock and, therefore, maintains full brightness. This part of the circuit operates regardless of whether any modified video options are selected.

We haven't forgotten SW1 and the other half of U5. They combine, along

with your system's clock, to produce the blank screen option mentioned earlier. When SW1 is high, your screen will not show any display. Video memory will still be updated, however, so that whenever SW1 is brought low the whole screen will be restored. This could be handy to do screen set-ups, hide your game moves in a two-player game, etc.

Table 1 offers a recap on the operation of switches SW1-SW4.

Table 1

SWITCH # MODE

1 2 3 4

H	X	X	X	BLANK SCREEN
L	L	X	X	NORMAL SCREEN
L	H	L	L	UPPER CASE ONLY
L	H	H	L	INVERSE UPPER CASE
L	H	L	H	DIM UPPER CASE
L	H	H	H	DIM INVERSE UPPER CASE

H = High, L = Low, X = Don't care

To test the modification, be sure all of the mode selection switches [SW1-SW4] are in the low state; this will ensure that you will have a normal screen to look at while you're setting up. We'll write a little program to fill the screen with mixed upper- and lower-case characters like the one below:

```
10 FORX=1TO12
20 PRINT"AaBbCcDdEeFfGgHhIiJj"
30 NEXT
```

This should fill your screen with alternating upper- and lower-case letters.

Using the mode selection switches, select inverse upper case; according to table 1 this should be L H H L. With the switches thus set, all lower-case letters should now be displayed as inverse upper case. Step through all the other modes to ascertain that they are working properly. If not, carefully check your wiring of both the circuit board and its interconnections to your C1P.

You may contact the authors at Orion Software Assocs., 147 Main St., P.O. Box 310, Ossining, NY 10562.

MICRO

AARDVARK

TRS-80 COLOR

OSI

VIC-64

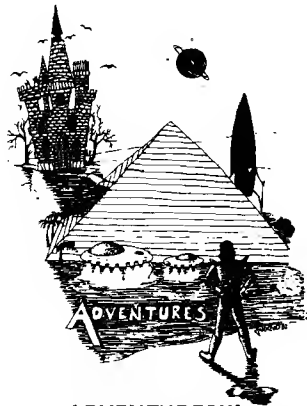
VIC-20

SINCLAIR

TIMEX



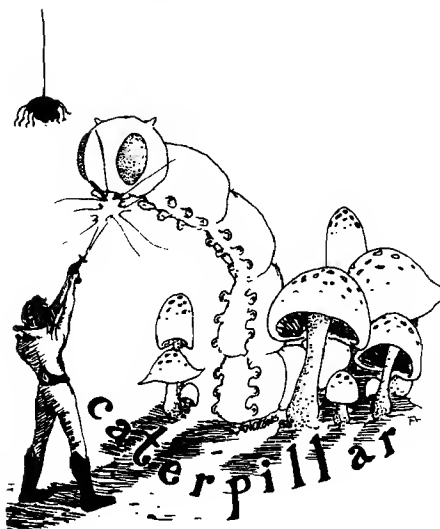
QUEST — A NEW IDEA IN ADVENTURE GAMES! Different from all the others. Quest is played on a computer generated map of Alesia. Your job is to gather men and supplies by combat, bargaining, exploration of ruins and temples and outright banditry. When your force is strong enough, you attack the Citadel of Moorlock in a life or death battle to the finish. Playable in 2 to 5 hours, this one is different every time. 16k TRS-80, TRS-80 Color, and Sinclair. 13K VIC-20. \$14.95 each.



ADVENTURES!!!

These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are interactive fantasies. It's like reading a book except that you are the main character as you give the computer commands like "Look in the Coffin" and "Light the torch.")

Adventures require 16k on TRS80, TRS80 color, and Sinclair. They require 8k on OSI and 13k on Vic-20. Derelict takes 12k on OSI. \$14.95 each.



CATERPILLAR

O.K., the Caterpillar does look a lot like a Centipede. We have spiders, falling fleas, monsters traipsing across the screen, poison mushrooms, and a lot of other familiar stuff. COLOR 80 requires 16k and Joysticks. This is Edson's best game to date. \$19.95 for TRS 80 COLOR.

PROGRAMMERS! SEE YOUR PROGRAM IN THIS SPACE!!

Aardvark traditionally pays the highest commissions in the industry and gives programs the widest possible coverage. Quality is the keyword. If your program is good and you want it presented by the best, send it to Aardvark.

ESCAPE FROM MARS

(by Rodger Olsen)

This ADVENTURE takes place on the RED PLANET. You'll have to explore a Martian city and deal with possibly hostile aliens to survive this one. A good first adventure.

PYRAMID (by Rodger Olsen)

This is our most challenging ADVENTURE. It is a treasure hunt in a pyramid full of problems. Exciting and tough!

HAUNTED HOUSE (by Bob Anderson)

It's a real adventure—with ghosts and ghouls and goblins and treasures and problems — but it is for kids. Designed for the 8 to 12 year old population and those who haven't tried Adventure before and want to start out real easy.

DERELICT

(by Rodger Olsen & Bob Anderson)

New winner in the toughest adventure from Aardvark sweepstakes. This one takes place on an alien ship that has been deserted for a thousand years — and is still dangerous!

Please specify system on all orders

ALSO FROM AARDVARK — This is only a partial list of what we carry. We have a lot of other games (particularly for the TRS-80 Color and OSI), business programs, blank tapes and disks and hardware. Send \$1.00 for our complete catalog.

AARDVARK - 80

2352 S. Commerce, Walled Lake, MI 48088

(313) 669-3110

Phone Orders Accepted 8:00 a.m. to 4:00 p.m. EST. Mon.-Fri.



TUBE FRENZY (by Dave Edson)

This is an almost indescribably fast action arcade game. It has fast action, an all new concept in play, simple rules, and 63 levels of difficulty. All machine code, requires Joysticks. Another great game by Dave Edson. TRS 80 COLOR ONLY. 16k and Joysticks required. \$19.95.



CATCH'EM

(by Dave Edson)

One of our simplest, fastest, funnest, all machine code arcade games. Raindrops and an incredible variety of other things come falling down on your head. Use the Joysticks to Catch'em. It's a BALL! — and a flying saucer! — and a Flying Y! — and so on. TRS 80 COLOR. \$19.95.

BASIC THAT ZOOOMMS!!

AT LAST AN AFFORDABLE COMPILER! The compiler allows you to write your programs in easy BASIC and then automatically generates a machine code equivalent that runs 50 to 150 times faster.

It does have some limitations. It takes at least 8k of RAM to run the compiler and it does only support a subset of BASIC—about 20 commands including FOR, NEXT, END, GOSUB, GOTO, IF, THEN, RETURN, END, PRINT, STOP, USR (X), PEEK, POKE, *, /, +, -, >, <, =, VARIABLE NAMES A-Z, SUBSCRIPTED VARIABLES and INTEGER NUMBERS FORM 0-64K.

TINY COMPILER is written in BASIC. I generates native, relocatable 6502 or 6801 code. It comes with a 20-page manual and can be modified or augmented by the user! \$24.95 on tape or disk for OSI, TRS-8 Color, or VIC.



Home Control Interface for C1P

by John Krout

A circuit is presented that uses the C1P's ACIA to control an ultrasonic transducer. The transducer generates signals that control the receiver modules.

BSR X-10 DRIVER

requires:

OSI C1P

BSR X-10

hardware modifications

Perhaps the greatest untapped potential of personal computers is control of common household devices such as lamps, air conditioners, and TV sets. A computer that turns an air conditioner off after you leave for work and on before you return will rapidly pay for itself in energy savings; and one that handles lights and entertainment equipment on a schedule will discourage burglars who prefer to enter unoccupied homes. You can probably think of more uses.

BSR markets the X-10 Control System through the mail and in Sears and Radio Shack stores. This remarkable system consists of a central command console about the size of a 3" x 5" file box, and up to 16 control modules, each the size of a pack of cigarettes. An appliance is plugged into a control module, which in turn is plugged into a power outlet. A control dial on each control module allows the user to set a unique unit code, ranging from 1 to 16, for that module. The user may control the module remotely via the console by pushing a button to specify the unit code. Another button turns the selected control module on or off.

A second form of control module includes a dimming control for lamps,

and a third form replaces a wall switch. Each control module is a radio receiver, which accepts transmitted commands only after receiving its own unit code. The command console is the transmitter, utilizing home power lines as an antenna.

Ohio Scientific was probably the first computer manufacturer to recognize the value of interfacing the X-10 command console to a personal computer. OSI now offers a hardware interface and a disk operating system to support the X-10. However, OSI charges a premium price for these items, and offers nothing to those using BASIC-in-ROM.

An optional feature of the command console provides the key to a simple and inexpensive interface to a computer. BSR also developed an ultrasonic hand-held command unit and combined the console with an ultrasonic receiver. This allows wireless control at a distance (like the ultrasonic hand-held TV controller). If you know the ultrasonic

code used by BSR, a few hardware modifications in your C1P will allow computer generation of the same codes, through an ultrasonic transducer, to transmit to the command console.

Figure 1 shows the various components of a single word of BSR code. The code is binary, with each bit represented by an 8-ms pattern of sound. A bit with value 1 is sent as 4 ms of tone followed by 4 ms of silence. A bit with value 0 is sent as 1.2 ms of tone followed by 6.8 ms of silence. The data word begins with a 1 bit, followed by five bits of data, followed by five inverted bits of the same data, and completed with 16 ms of tone and 24 ms of silence. The tone itself is 40 KHz. The five-bit code for each control module and function is shown in table 1.

A single latched output bit in the computer is all you need to transmit the code. The C1P uses latched output bits to scan the keyboard and joysticks as well as drive a digital-to-analog converter (D/A) circuit. However, BASIC

Figure 1

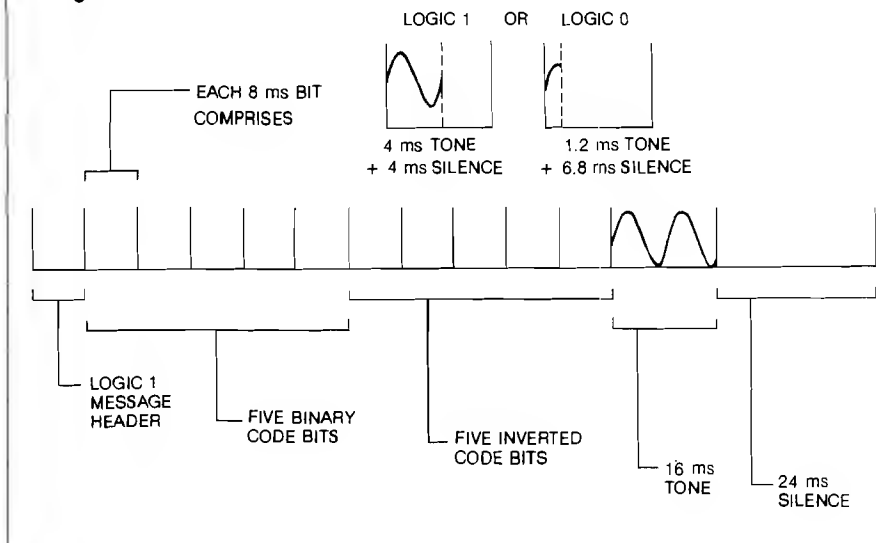


Figure 2

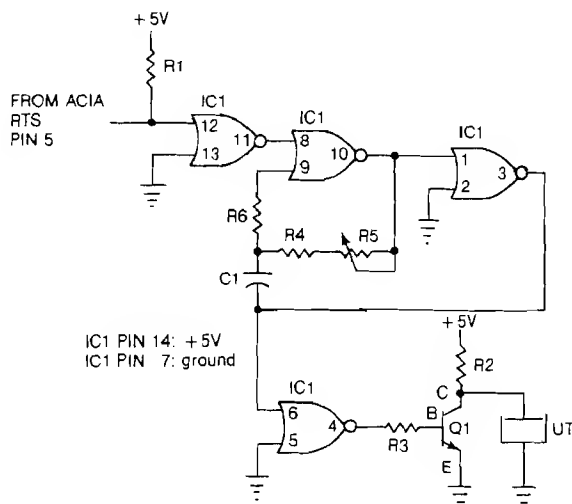


Table 2

Item	Value
IC1	4001 CMOS quad NOR gate 14-pin DIP
R1	2.2K resistor
R2	2.2K resistor
R3	2.2K resistor
R4	12K resistor
R5	50K trim potentiometer
R6	330K resistor
C1	330 pF capacitor
Q1	Sylvania ECG123A transistor or equivalent
UT	40 KHz ultrasonic transducer

Table 1

Unit Code	Binary Code
1	0 1 1 0 0
2	1 1 1 0 0
3	0 0 1 0 0
4	1 0 1 0 0
5	0 0 0 1 0
6	1 0 0 1 0
7	0 1 0 1 0
8	1 1 0 1 0
9	0 1 1 1 0
10	1 1 1 1 0
11	0 0 1 1 0
12	1 0 1 1 0
13	0 0 0 0 0
14	1 0 0 0 0
15	0 1 0 0 0
16	1 1 0 0 0

Function Code	Binary Code
17/All Units Off	0 0 0 0 1
18/All Lights On	0 0 0 1 1
19/On	0 0 1 0 1
20/Off	0 0 1 1 1
21/Dim	0 1 0 0 1
22/Bright	0 1 0 1 1

continually scans the keyboard (unless the Control-C break is disabled by an appropriate POKE) so some sort of tone is almost always being produced on the D/A output while BASIC, or any other keyboard-oriented program, is being used. This makes using the D/A unpleasant for music composition and playback.

A less well-known bit of latched output exists in the C1P. This is the RTS [Request-To-Send] line associated with

the 6850 Asynchronous Serial Communications Interface chip (ACIA) used in the C1P to exchange data with a cassette machine, modem, or printer. This particular line is not used by the C1P, although the ACIA designers provide it so that a computer can indicate whether or not it is ready to receive data.

The control register of the ACIA chip controls the status of the RTS line, among other ACIA activities. In BASIC, whenever the Break key is depressed, the control register is reset to a value of 17 and RTS goes low. If you POKE a value of 64 to the register, then RTS will go high and stay there until another value is stored in the register. One advantage of this bit in the BSR interface is that it will automatically turn off when Break is depressed. The ACIA control register is located in the C1P at

address 61440 (\$F000).

The RTS line can be toggled at a 40-KHz rate to produce the BSR code. Since the C1P uses a standard clock rate of 1 MHz, the wavelength of a 40-KHz tone is precisely 25 clock cycles. However, I found by timing my C1P with an oscilloscope that its clock is running about 4% slow. Thus, I could produce the tone using a 24-clock cycle wavelength. Instead, I chose to build a free-running 40-KHz oscillator and use the RTS line to switch the oscillator output to an ultrasonic transducer.

The oscillator circuit is shown in figure 2, and the parts are listed in table 2. The only part not universally available is the ultrasonic transducer, a capacitive loudspeaker that creates the actual tone. Since these devices are

Listing 1

```

10 ; ASSEMBLY LISTING OF BSR X-10 DRIVER ROUTINE
20 ;
30 ; BY JOHN KROUT
40 ;
50     *=$0222
60     DELAY=$FC91
70 ;
80 START JSR $AE05 ; puts argument in $AE,AF
90     LDX $AF
100    LDA TABLE-1,X
110    STA $AF ; lookup & store code word
120    LDA #5
130    STA $15
140 MASTER JSR WORD
150    DEC $15 ; counts data words sent
160    BNE MASTER
170    RTS ; return to Basic
180 ;
190 ;
200 WORD JSR LOGIC1 ; send message header bit
210    LDA $AF ; command code into accumulator
220    JSR SEND ; send top 5 accumulator bits
230    LDA $AF ; reload accumulator

```

(continued)

Listing 1 (continued)

```

240     EOR #255 ; invert accumulator bits
250     JSR SEND ; send 5 inverted bits
260     LDA #64
270     STA $F000 ; begin 16 ms tone
280     LDX #4
290     STX $16
300 LOOP1 JSR MS4
310     DEC $16
320     BNE LOOP1
330     LDA #17
340     STA $F000 ; begin 24 ms silence
350     LDX #5
360     STX $16
370 LOOP2 JSR MS4
380     DEC $16
390     BNE LOOP2
400     JMP MS4
410 ;
420 SEND  STA $13
430     LDA #5
440     STA $14 ; counter for bits sent
450 ROLL  ROL $13 ; place bit in Carry
460     BCC ZERO ; branch if Carry=0
470     JSR LOGIC1 ; send logic 1
480     JMP COUNT
490 ZERO  JSR LOGIC0 ; send logic 0
500 COUNT DEC $14
510     BNE ROLL ; branch until 5 bits sent
520     RTS
530 ;
540 LOGIC1 LDA #64
550     STA $F000 ; begin 4 ms tone
560     JSR MS4
570     LDA #17
580     STA $F000 ; begin 4 ms silence
590     JMP MS4
600 ;
610 LOGIC0 LDA #64
620     STA $F000 ; begin 1.2 ms tone
630     JSR MS1.2
640     LDA #17
650     STA $F000 ; begin 6.8 ms silence
660     JMP MS6.8
670 ;
680 MS4    LDX #15
690 LOOP3  DEX
700     BNE LOOP3
710     LDX #3
720     JMP DELAY
730 ;
740 MS1.2  LDX #228
750 LOOP4  DEX
760     BNE LOOP4
770     RTS
780 ;
790 MS6.8  LDX #52
800 LOOP5  DEX
810     BNE LOOP5
820     LDX #5
830     JMP DELAY
840 ;
850 TABLE .BYTE 96,224,32,160,16,144,80,208
860     .BYTE 112,240,48,176,0,128,64,192
870     .BYTE 8,24,40,56,72,88

```

Listing 2

```

FC91 A0F8 LDY ##FB
FC93 88 DEY
FC94 D0FD BNE $FC93
FC96 55FF EOR $FF,X
FC98 CA DEX
FC99 D0F6 BNE $FC91
FC9B 60 RTS

```

Listing 3

```

100     *=$0222
110 START LDX #64
120     STX $F000
130     NOP
140     LDX #198
150 X1    DEX
160     BNE X1
170     STX $F000
180     LDX #3
190     LDX #198
200 X2    DEX
210     BNE X2
220     JMP START

```

pretuned to a specific frequency, be sure the one you buy is set to 40 KHz. One transducer that costs less than \$10 is #J4-815 in the Calectro catalog.

The circuit can be installed on any of the unconnected prototype sockets adjacent to the ACIA, with a pair of output lines running out of the computer case to the transducer. Or the circuit can be placed externally on perf-board, with connection lines for power, ground, and RTS. Because my C1P board is crowded with add-ons, I chose the latter method. I recommend that you do not mount the transducer to the C1P case because it has to be in a fairly direct line with the receiver microphone grid on the front face of the command console for transmission to be reliable. To preserve aiming flexibility, put the transducer on a lengthy flexible signal cable. You can secure it to the command console grid, if you wish.

A USR software-driver routine for the interface appears in listing 1. This routine begins by calling the ROM BASIC subroutine at address \$AE05, which deciphers the argument value within the parentheses following the USR call in BASIC text, and puts that value in locations \$AE and \$AF in the form of a 15-bit integer with a sign bit. Any argument value outside the range of -32768 to +32767 will cause a function call error if the \$AE05 routine is called.

The USR routine assumes that the argument is a number between 1 and 22, corresponding to a BSR unit or command number. Lines 90 through 110 look up the appropriate five-bit command code in a data table and replace the original argument value with the code. Lines 120 through 160 produce five repetitions of code transmission, a factor which was found reliable when used in a BASIC program that turned house lights on and off over a two-hour period. This means that each USR call takes about 640 ms.

The main subroutine WORD begins at line 200 with transmission of the single-bit prefix, a logic 1. Then the command code is loaded and transmitted once, reloaded, inverted in line 240, and transmitted again. The code-word suffix is sent by the remainder of WORD.

Subroutine SEND analyzes each bit of the five-bit command code and transmits the appropriate tone sequence. In line 450, ROL \$13 places each command bit into the Carry bit of the 6502

status register and, in line 460, BCC branches if the Carry bit is zero.

Subroutine LOGIC1 turns on the RTS line, waits 4 ms, turns off the RTS line, and waits another 4 ms. LOGIC0 waits 1.2 ms after turning on RTS and then waits 6.8 ms after turning off RTS.

The three timing subroutines MS4, MS1.2, and MS6.8 handle the precise waiting periods required by the other subroutines. Each includes a DEX/BNE loop that takes five clock cycles per iteration, except that only four are used when BNE does not branch. The prior LDX immediate in each case takes two cycles, as does the following LDX immediate in MS4 and MS6.8. These two routines then use three cycles to JMP to a routine called DELAY in the monitor ROM at \$FC91.

Delay is a time-delay loop that, perhaps, was included in ROM to aid in disk I/O. It appears in listing 2 and uses 1250 cycles per iteration, with the number of repetitions controlled by the 6502 X register. The RTS at the end takes an extra six cycles. The difficulty with DELAY is that it wipes out not only the X and Y registers but also the

Listing 4

```
10 PRINT"Enter your C1P clock"
15 PRINT"rate as a decimal frac-"
20 PRINT"tion of the standard 1"
25 PRINT"megahertz clock rate"
30 PRINT"example: 6% fast is"
35 PRINT"entered as 1.06";
40 INPUT Q
45 M4=INT(4000*Q)-12
50 M1=INT(1200*Q)-7
55 M6=INT(6800*Q)-12
60 D=1250
65 D4=INT(M4/D):R4=INT((M4-D4*D)/5)
70 R1=INT(M1/5)
75 D6=INT(M6/D):R6=INT((M6-D6*D)/5)
80 POKE675,R4:POKE680,D4
85 POKE685,D1
90 POKE691,R6:POKE696,D6
```

Listing 5

```
5 X=546:Z=60000
7 SAVE
9 PRINT:PRINT
10 FORI=0TO175
20 IFI=INT(I/15)*15THENPRINT:
PRINTZ;"DATA";Z=Z+5:GOTO30
25 PRINT";";
30 A$=STR$(PEEK(I+X)):PRINTRIGHT$(A$,LEN(A$)-1);
40 NEXT
50 PRINT
60 PRINT"20 POKE11,34:POKE12,3"
70 PRINT"30 FORI=0TO175:READA:
POKEI+546,A:NEXT"
80 PRINT"40 NEW"
90 PRINT"POKE515,0:RUN"
95 POKE517,0
```

Listing 6

```
60000 DATA32,5,174,166,175,189,187,2,133,175,169,5,133,21,32
60005 DATA56,2,198,21,208,249,96,32,130,2,165,175,32,106,2
60010 DATA165,175,73,255,32,106,2,169,64,141,0,240,162,4,134
60015 DATA22,32,162,2,198,22,208,249,169,17,141,0,240,162,5
60020 DATA134,22,32,162,2,198,22,208,249,76,162,2,133,19,169
60025 DATA5,133,20,38,19,144,6,32,130,2,76,125,2,32,146
60030 DATA2,198,20,208,239,96,169,64,141,0,240,32,162,2,169
60035 DATA17,141,0,240,76,162,2,169,64,141,0,240,32,172,2
60040 DATA169,17,141,0,240,76,178,2,162,15,202,208,253,162,3
60045 DATA76,145,252,162,228,202,208,253,96,162,52,202,208,253,162
60050 DATA5,76,145,252,96,224,32,160,16,144,80,208,112,240,48
60055 DATA174,0,128,64,192,0,24,40,56,72,88
20 POKE11,34:POKE12,2
30 FORI=0TO175:READA:POKEI+546,A:NEXT
40 NEW
POKE515,0:RUN
```



Plus Sensational Limited-Time Savings On Ohio Scientific C1P Series personal computers, Superboard and C1P accessories, spare replacement parts, printers, monitors, integrated circuits, and other computer-related components.

To Order

Call us directly or return order coupon with your check, money order, or Mastercard or Visa Account Number. Orders will normally be shipped within 48 hours after receipt. \$100.00 minimum order.

FREE

Sampler Cassettes with each Superboard II and C1P series order!

Taxi (Game), Electronic Equations, Loan Finance, Straight and Constant Depreciation, Uneven Cash Flows

Tiger Tank, Flip Flop, (Logic Game), Hectic, Black Jack, Master Mind

Super Sale!

**NOW!!
\$149.95!!**

60% 40% Off On Ohio Scientific Superboard II A Complete Computer System On A Board

Includes full-size 53-key keyboard; video and audio cassette interfaces; SWAP, Modem, sampler cassettes; manual; 8K BASIC-in-ROM, with 8K RAM. Requires 5-V/3 amp regulated DC power supply. 30-day limited warranty. Supply is limited. ~~ONLY \$200.00~~ NOW! \$149.95!!



Cleveland Consumer Computers & Components

1333 S. Chillicothe Road, Aurora, OH 44202
TO ORDER: CALL 1-800-321-5805 TOLL FREE
(Ohio Residents Call 216-562-4136)

\$149.95

☐ SUPERBOARD II, ~~\$200.00~~

☐ Send Detailed Catalog/Order Form

Name

Address

City State Zip

Payment by enclosed check or money order or charge to:

☐ MasterCard

☐ VISA

Account # Expiration Date

Total Amount Charged or Enclosed \$

Ohio Residents Add 5.5% Sales Tax. All Orders Will Be Shipped Insured By UPS Unless Requested Otherwise.

Listing 7

```

5 GOTO2000
6 :
10 REM ... LITESHOW CONTROL PROGRAM ...
12 REM ... FOR BSR X-10 INTERFACE ...
14 REM ... BY JOHN KROUT
99 :
100 REM SPOTS: 1 ON, 1 OFF
101 :
110 FORA=1TO3:B=A+1:IFA=3THENB=1
120 Y=USR(B):IFPEEK(Q)=EGOTO1000
130 Y=USR(A):IFPEEK(Q)=EGOTO1000
140 NEXT:GOTO110
199 :
200 REM SPOTS: 2 ON, 1 OFF
201 :
210 FORA=1TO3
220 Y=USR(18):IFPEEK(Q)=EGOTO1000: REM ALL SPOTS ON
230 Y=USR(A):Y=USR(20):IFPEEK(Q)=EGOTO1000: REM 1 OFF
235 FORI=1TO1000:NEXT:REM TIME DELAY
240 NEXT:GOTO210
299 :
300 REM KEYBOARD CONTROL
301 :
302 GOSUB4000:PRINT" SPOTS":PRINT:PRINT"STROBES":PRINT:PRINT"PROJECTOR"
304 POKEG,89:POKEG+2,66:POKEG+4,82
310 POKES30,1:POKE57088,127:P=PEEK(57088):POKE530,0
315 IFPEEK(Q)=EGOTO1000
320 FORA=1TO7:IFS(A,1)=PGOTO335
325 NEXT:GOTO310
335 Y=USR(A):IFS(A,0)=0THENY=USR(19):S(A,0)=1:POKE5(A,2),43:GOTO310
340 Y=USR(20):S(A,0)=0:POKE5(A,2),32:GOTO310
399 :
400 REM STROBES: 1 ON, 1 OFF
401 :
410 FORA=4TO6:B=A+1:IFA=6THENB=4
420 Y=USR(B):Y=USR(19):IFPEEK(Q)=EGOTO1000
430 Y=USR(A):Y=USR(20):IFPEEK(Q)=EGOTO1000
440 NEXT:GOTO410
499 :
1000 REM MAIN MENU
1020 FORI=1TO7:S(I,0)=0:NEXT:REM STATUS RESET
1025 GOSUB4000
1030 PRINT"MAIN MENU":PRINT
1040 PRINT"1. SPOTS: 1 ON, 1 OFF":PRINT:PRINT
1042 PRINT"2. SPOTS: 2 ON, 1 OFF":PRINT:PRINT
1044 PRINT"3. KEYBOARD CONTROL":PRINT:PRINT
1046 PRINT"4. STROBES: 1 ON, 1 OFF":PRINT:PRINT
1100 INPUT"function number":F:PRINT
1110 IFF(1ORF)10ORF:INT(F)GOTO1100
1115 Y=USR(17): REM SHUTDOWN
1120 ONF6TO100,200,300,400
1200 END
2000 REM INIT
2010 DIMS(7,2)
2020 S(1,1)=127
2030 S(2,1)=191
2040 S(3,1)=223
2050 S(4,1)=239
2060 S(5,1)=247
2070 S(6,1)=251
2080 S(7,1)=253
2100 Q=57100:E=222
2110 G=53901
2120 S(1,2)=G+64
2130 S(2,2)=G+66
2140 S(3,2)=G+68
2150 S(4,2)=G+128
2160 S(5,2)=G+130
2170 S(6,2)=G+132
2180 S(7,2)=G+194
2999 GOTO1000
4000 REM SCREEN CLR SUB
4010 FORI=1TO28:PRINT:NEXT:RETURN

```

accumulator. The latter could have been avoided by using a few NOPs instead of the EOR. In the USR routine, whenever a delay routine is called, this problem forces storage in memory of the command word, the number of

words sent, and the number of bits sent. Since BASIC does not use the input buffer beginning at \$13 for anything other than input, USR can access that space with compact and speedy page zero addressing for data storage on a

non-permanent basis. Alternatives include stack storage and replacing DELAY with your own non-destructive time delay.

Because my C1P runs about 4% slow, the time delays in MS4, MS6.8, MS1.2, and the message suffix portion of WORD have been shortened about 4% to compensate. If you can obtain an oscilloscope, listing 3 will load and execute a useful infinite loop USR routine. This routine turns on RTS for precisely 999 cycles, and then turns off RTS for 1001 cycles, giving an overall wavelength of exactly 2 ms for a machine running at exactly 1 MHz. If your machine is running a few percent slow or fast, listing 4 will compute and POKE the necessary loop constant alterations to the BSR X-10 driver routine.

As with many USR routines, it is convenient to place the driver in unused memory below BASIC text, starting at \$0222. Because the OSI Assembler occupies this space and cannot directly assemble the routine there, a loader in BASIC is useful. Listing 6 uses the familiar method of POKEing numbers from DATA statements to memory, and is itself a product of listing 5, a BASIC program generator. Listing 5 includes the very advantageous features of placing two immediate-mode commands at the end of listing 6: a POKE to terminate LOAD, and RUN. Since the DATA statements are so long in this case, the NEW statement in line 40 of listing 6 erases listing 6 after its work is done, leaving behind the driver routine and the data in locations 11 and 12 that tell BASIC where the USR routine begins.

Listing 7 is a BASIC light show control program, which is loaded after listing 6 has finished. The program presumes that X-10 lamp modules 1, 2, and 3 control colored spotlights, that appliance modules 4, 5, and 6 control colored strobe lights, and that appliance module 7 controls the lamp of a slide projector. Projector lamps usually exceed 300 watts. You should keep the projector fan running even when the lamp is off to cool the lamp and avoid a blowout.

Would you like some automation in your life? Perhaps you need a timer for your toaster, or a security system for your office copier. Computer intelligence plus BSR X-10 versatility can do it for you.

The author may be contacted at 5108 N. 23rd Rd., Arlington, VA 22207.

MICRO

ATARI Meets the BSR X-10

by David A. Hayes

A circuit is presented to interface the ultrasonic version of the BSR X-10 home control system to Atari computers. Programming information and a sample program are included.

Demo Program requires:

Atari 400/800
BSR X-10

To use the BSR X-10 home control device, many computers require a hardware modification. David Staehlin presented a circuit, in the January 1982 issue of *BYTE* magazine, which will couple a non-ultrasonic BSR X-10 to an RS-232 port. I have interfaced the Atari's controller jack port to the more common ultrasonic version of the BSR X-10. Figure 1 shows the complete interface circuit required for this purpose. Modification of the BSR X-10 is not trivial and should be performed by competent technicians only.

The program in listing 1 loads a machine-language program into page 6 of memory. Line 100 sets up controller jack 1, pin 1, as output. Table 1 lists the code that the BSR X-10 understands. The machine-language program sends this code out controller jack 1, pin 1, whenever it is called by the *USR* routine.

For example, if you have made the appropriate hardware modifications, have typed in the program in listing 1, and now want to turn all lights on, line 110 of your program should look like this:

```
110 X = USR(1536,0,0,0,128,128,
          128,128,128,0,0)
```

Now turn on channel five.

```
120 X = USR(1536,0,0,0,128,0,128,
          128,128,0,128):REM SELECT
          CHANNEL 5
130 X = USR(1536,0,0,128,0,128,128,
          128,0,128,0):REM TURN ON
```

The author may be contacted at 2004 Woody Drive, Kingston, TN 37763.

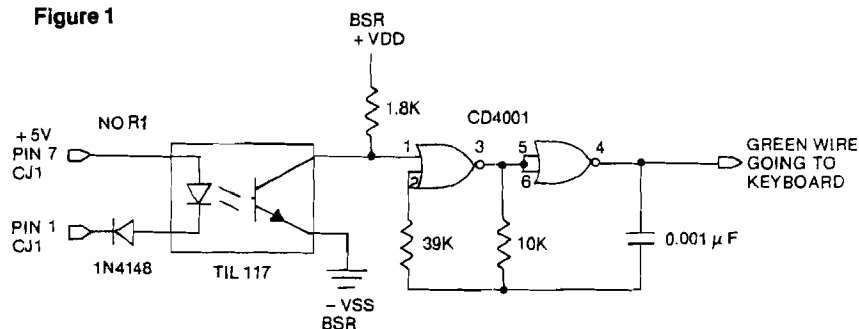
(Continued on next page)

Table 1

FUNCTION	X = USR(1536,A,B,C,D,E,F,G,H,I,J)
ALL LIGHTS ON	0,0,0,128,128,128,128,128,0,0
ALL OFF	0,0,0,0,128,128,128,128,128,0
ON	0,0,128,0,128,128,128,0,128,0
OFF	0,0,128,128,128,128,128,0,0,0
BRIGHTEN	0,128,0,128,128,128,0,128,0,0
DIM	0,128,0,0,128,128,0,128,128,0

CHANNEL	
1	0,128,128,0,0,128,0,0,128,128
2	128,128,128,0,0,0,0,0,128,128
3	0,0,128,0,0,128,128,0,128,128
4	128,0,128,0,0,0,128,0,128,128
5	0,0,0,128,0,128,128,128,0,128
6	128,0,0,128,0,0,128,128,0,128
7	0,128,0,128,0,128,0,128,0,128
8	128,128,0,128,0,0,0,128,0,128
9	0,128,128,128,0,128,0,0,0,128
10	128,128,128,128,0,0,0,0,0,128
11	0,0,128,128,0,128,128,0,0,128
12	128,0,128,128,0,0,128,0,0,128
13	0,0,0,0,0,128,128,128,128,128
14	128,0,0,0,0,0,128,128,128,128
15	0,128,0,0,0,128,0,128,128,128
16	128,128,0,0,0,0,0,128,128,128

Figure 1



Listing 1

```

10 FOR ADD=1536 TO 1756: READ INST: POKE ADD,INST: NEXT ADD
20 DATA 104,32,138,6,104,104,48,6,32,169,6,76,17,6,32,138,6,
    104,104,48,6,32,169
25 DATA 6,76,30,6,32,138,6,104,104
30 DATA 48,6,32,169,6,76,43,6,32,138,6,104,104,48,6,32,169,
    6,76,56,6,32,138,6
35 DATA 104,104,48,6,32,169,6,76,69
40 DATA 6,32,138,6,104,104,48,6,32,169,6,76,82,6,32,138,6,
    104,104,48,6,32,169
45 DATA 6,76,95,6,32,138,6,104,104
50 DATA 48,6,32,169,6,76,108,6,32,138,6,104,104,48,6,32,
    169,6,76,121,6,32,138
55 DATA 6,104,104,48,6,32,169,6,76
60 DATA 134,6,32,138,6,32,200,6,96,169,254,141,0,211,162,
    120,160,10,136,208
65 DATA 253,202,208,248,169,255,141,0,211,162
70 DATA 120,160,10,136,208,253,202,208,248,96,169,254,141,
    0,211,162,40,160,10
75 DATA 136,208,253,202,208,248,169,255,141
80 DATA 0,211,162,31,160,70,136,208,253,202,208,248,96,169,
    254,141,0,211,162
85 DATA 54,160,70,136,208,253,202,208,248
90 DATA 169,255,141,0,211,96
100 POKE 54018,56: POKE 54016,1: POKE 54018,60: POKE 54016,1

```

MICRO™

OHIO SCIENTIFIC

NEW PROGRAMS!

SCOUT—Full color, machine language, fast action and graphics! After a year of development, comes the all machine language **SCOUT**. Patrol the planet surface protecting and saving the human population from abductors. Turn your OSI into a real arcade!

\$24.95 C4PMF, C8PDF.

Send for our FREE catalog. We have what you want for less: S-FORTH \$39, FULL SCREEN EDITOR \$19, ADVENTURE \$19, SKYHAWK \$8, TOUCH TYPING \$19, INTELLIGENT TERMINAL \$24, THE WIZARD'S CITY \$12, UTILITIES, and much more for the C1P to the C8PDF.

(312) 259-3150

AURORA SOFTWARE



37 S. Mitchell
Arlington Heights,
Illinois 60005



OSI Disk Users

Double your disk storage capacity Without adding disk drives

Now you can more than double your usable floppy disk storage capacity—for a fraction of the cost of additional disk drives. Modular Systems' DiskDoubler™ is a double-density adapter that doubles the storage capacity of each disk track. The DiskDoubler plugs directly into an OSI disk interface board. No changes to hardware or software are required.

The DiskDoubler increases total disk space under OS-65U to 550K; under OS-65D to 473K for 8-inch floppies, to 163K for mini-floppies. With the DiskDoubler, each drive does the work of two. You can have more and larger programs, related files, and disk utilities on the same disk—for easier operation without constant disk changes.

Your OSI system is an investment in computing power. Get the full value from the disk hardware and software that you already own. Just write to us, and we'll send you the full story on the DiskDoubler, along with the rest of our growing family of products for OSI disk systems.

™DiskDoubler is a trademark of Modular Systems.



Post Office Box 16C
Oradell, NJ 07649-0016
Telephone 201 262.0093

APPLE II PERIPHERAL DEVELOPERS:

Your complex function prototype requires
the best wirewrap board available.

SPECTRUM SYSTEMS MAKES IT!

Fully Extended Wirewrap Protoboard.

Size: 2.8 by 10.7 inch 2 layer PC.

Capacity: up to 58-16 pin or 12-40 pin
or any combination sockets inbetween.

Carefully designed +5 and GND planes provide for
the minimum electrical noise, low impedance, hi
capacitance, and maximum versatility in the layout
of IC's, capacitors, discretes and I/O connectors.

Wire-wrap technique documentation included.

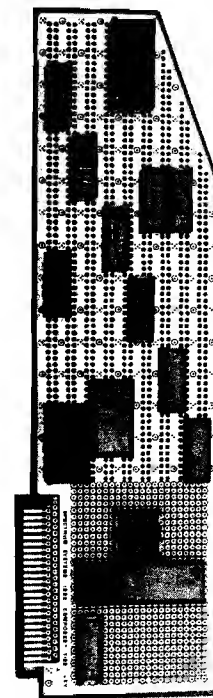
Terms:

- \$45.00 + (6% Cal. Res. tax) + \$2.00 S&H.
- All payments must be in U.S. funds drawn on a U.S. bank.
- Outside U.S. add 10%.
- Cashier check/money order allow 30 day ARO.
- Personal checks add 2 weeks.
- No credit cards or cash, Please!

Spectrum Systems

P.O. Box 2262

Santa Barbara, Ca. 93120

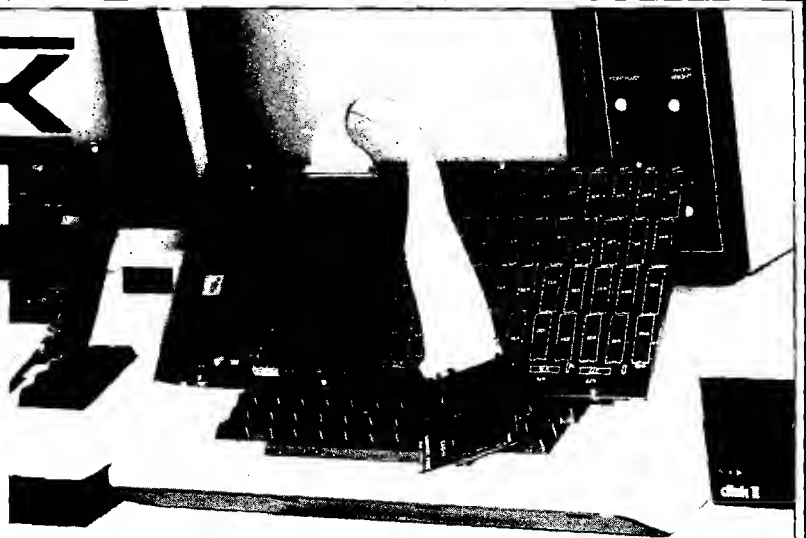


Apple II is a trademark of Apple Computers

DTACK

10

The 68000 DREAM MACHINE



SISE p. 86

WE (SORT OF) LIED:

Motorola has been promoting its advanced microprocessor chip as a vehicle for large, complex systems **exclusively**. Now, the 68000 does work well as the heart of big, complex systems. But their promotional literature implies that one can **only** build big, complex systems with the 68000, and that is dead wrong (in our opinion). Nevertheless, the public (that's you!) perception of the 68000 follows Motorola's line: **Big systems. Complex systems.**

Our boards are **not** complex and not necessarily big (starting at 4K). Our newsletter is subtitled "The Journal of Simple 68000 Systems." But since the public has become conditioned to the 68000 as a vehicle for FORTRAN, UNIX, LISP, PASCAL and SMALLTALK people naturally expect all these with our \$595 (starting price) simple attached processor. **Wrong!**

We wrote our last ad to **understate** the software we have available because we wanted to get rid of all those guys who want to run (multi-user, multi-tasking) UNIX on their Apple II and two floppy disks. Running UNIX using two 143K floppies is, well, absurd. The utilities alone require more than 5 megabytes of hard disk.

HERE'S THE TRUTH:

We do have some very useful 68000 utility programs. One of these will provide, in conjunction with a suitable BASIC compiler such as PETSPEED (Pet/CBM) or TASC (Apple II), a five to twelve times speedup of your BASIC program. If you have read a serious compiler review, you will have learned that compilers cannot speed up floating point operations (especially transcendental). Our board, and the utility software we provide, **does** speed up those operations.

Add this line in front of an Applesoft program:

```
5 PRINT CHR$(4);"BLOADUTIL4,A$8600":SYS38383
```

That's all it takes to link our board into Applesoft (assuming you have Applesoft loaded into a 16K RAM card). Now run your program as is for faster number-crunching or compile it to add the benefit of faster "interpretation". Operation with the Pet/CBM is similar.

68000 SOURCE CODE:

For Apple II users only, we provide a nearly full disk of **unprotected** 68000 source code. To use it you will have to have DOS toolkit (\$75) and ASSEM68K (\$95), both available from third parties. Here's what you get:

1) 68000 source code for our Microsoft compatible floating point package, including LOG, EXP, SQR, SIN, COS, TAN, ATN along with the basic four functions. The code is set up to work either linked into BASIC or with our developmental HALGOL language. 85 sectors.

2) 68000 source code for the PROM monitor. 35 sectors.

3) 68000 source code for a very high speed interactive 3-D graphics demo. 115 sectors.

4) 68000 source code for the HALGOL threaded interpreter. Works with the 68000 floating point package. 56 sectors.

5) 6502 source code for the utilities to link into the BASIC floating point routines and utility and debug code to link into the 68000 PROM monitor. 113 sectors.

The above routines almost fill a standard Apple DOS 3.3 floppy. We provide a second disk (very nearly filled) with various utility and demonstration programs.

SWIFTUS MAXIMUS:

Our last advertisement implied that we sold 8MHz boards to hackers and 12.5MHz boards to businesses. That was sort of true because when that ad was written the 12.5MHz 68000 was a very expensive part (list \$332 ea). Motorola has now dropped the price to \$111 and we have adjusted our prices accordingly. So now even hackers can afford a 12.5MHz 68000 board. With, we remind you, **absolutely zero wait states**.

'Swiftus maximus'? Do you know of any other microprocessor based product that can do a 32 bit add in 0.48 microseconds?

AN EDUCATIONAL BOARD?

If you want to learn how to program the 68000 at the assembly language level there is no better way than to have one disk full of demonstration programs and another disk full of machine readable (and user-modifiable) 68000 source code.

Those other 'educational boards' have 4MHz clock signals (even the one promoted as having a 6MHz CPU, honest!) so we'll call them **slow learners**. They do not come with any significant amount of demo or utility software. And they communicate with the host computer via RS 232, 9600 baud max. That's 1K byte/sec. Our board communicates over a parallel port with hardware AND software handshake, at 71K bytes/sec! We'll call those other boards **handicapped learners**.

Our board is definitely not for everyone. But some people find it very, very useful. Which group do you fit into?

DIGITAL ACOUSTICS
1415 E. McFadden, Ste. F
Santa Ana, CA 92705
(714) 835-4884

68000 Logic Instructions

by Joe Hootman

This is the third in a series of articles on programming the 68000. Professor Hootman is presenting the instruction set of the 68000 microprocessor and will then consider the addressing modes and how they apply to the various instructions. This month's topic is the logical instructions.

The logic instructions implemented in the 68000 are given in table 1. These instructions are the AND, the OR, the NOT, and the EOR. The implementation of the logical operations is straightforward. The logic operations affect the CCR depending on the results of the operation. It should be noted that the logical operations do not operate on the address registers directly.

The logic operations on the status register are privileged. Logical operations on the user condition code register are not privileged.

Joe Hootman can be contacted at the University of North Dakota, Department of Electrical Engineering, University Station, Grand Forks, North Dakota 58202.

Table 1: Logic Instructions

Mnemonic	Data Size/CCR	Name	Comments																																																																			
AND	8, 16, 32 CCR X N Z V C * * * 0 0	Logical AND	<p>The source and destination are logically ANDed and the result stored in the destination.</p> <p>Opword Format</p> <table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>Register</td><td>Op Mode</td><td colspan="2">Effective Address</td><td>Mode</td><td>Register</td><td colspan="6"></td></tr></table> <p>Register — Any of the eight data registers.</p> <p>Op Mode field</p> <table><tr><td>Byte</td><td>Word</td><td>Long word</td></tr><tr><td>A) 000</td><td>001</td><td>010</td></tr></table> <p>Data register ANDed with the EA and result left in the data register.</p> <table><tr><td>B) 100</td><td>101</td><td>110</td></tr></table> <p>EA ANDed with the data register and result left in the EA.</p> <p>For case A of the Op Modes the following effective addressing modes cannot be used: 2, 13, 14.* For case B of the Op Modes the following effective addressing modes cannot be used: 1, 2, 10, 11, 12, 13, 14.*</p>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	1	0	0	Register	Op Mode	Effective Address		Mode	Register							Byte	Word	Long word	A) 000	001	010	B) 100	101	110																										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																							
1	1	0	0	Register	Op Mode	Effective Address		Mode	Register																																																													
Byte	Word	Long word																																																																				
A) 000	001	010																																																																				
B) 100	101	110																																																																				
ANDI	8, 16, 32 CCR X N Z V C * * * 0 0	AND Immediate	<p>The immediate data and the destination are logically ANDed and the result stored in the destination.</p> <p>Opword Format</p> <table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>Size</td><td colspan="2">Effective Address</td><td>Mode</td><td>Register</td><td colspan="3"></td></tr><tr><td colspan="8">Word data (16 bits including the first 8 bits)</td><td colspan="8">Byte data (8 bits)</td></tr><tr><td colspan="16">Long data (32 bits including the previous bits)</td></tr></table> <p>Size field</p> <table><tr><td>00 - Byte</td></tr><tr><td>01 - Word</td></tr><tr><td>10 - Long word</td></tr></table> <p>The following addressing modes cannot be used: 2, 10, 11, 12, 13, 14.*</p>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0	1	0	Size	Effective Address		Mode	Register				Word data (16 bits including the first 8 bits)								Byte data (8 bits)								Long data (32 bits including the previous bits)																00 - Byte	01 - Word	10 - Long word
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																							
0	0	0	0	0	0	1	0	Size	Effective Address		Mode	Register																																																										
Word data (16 bits including the first 8 bits)								Byte data (8 bits)																																																														
Long data (32 bits including the previous bits)																																																																						
00 - Byte																																																																						
01 - Word																																																																						
10 - Long word																																																																						
ANDI to CCR	8 CCR X N Z V C * * * * *	AND Immediate to Condition Code Register	<p>The immediate data is ANDed with the CCR and the results stored in the CCR. The state of the CCR after the operation depends on the previous data in the CCR and the immediate data in the operation.</p> <p>Opword Format</p> <table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td colspan="8">Byte Data</td><td colspan="8"></td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0	Byte Data																																		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																							
0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0																																																							
Byte Data																																																																						

(continued)

Table 1 (continued)

Mnemonic	Data Size/CCR	Name	Comments																																			
EOR	8, 16, 32 CCR X N Z V C - * * 0 0	Exclusive OR Logical	The source and the destination are exclusively ORed together and the result stored in the destination. (Data registers only for source data.) Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 <table><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>Register</td><td>Op Mode</td><td>Effective Address Mode</td><td>Register</td></tr></table> Register field — Any one of the eight data registers can be specified. Op Mode field 100 - Byte 101 - Word 110 - Long word The effective address specifies the destination of the result of the operation and the following addressing modes cannot be used: 2, 10, 11, 12, 13, 14.*	1	0	1	1	Register	Op Mode	Effective Address Mode	Register																											
1	0	1	1	Register	Op Mode	Effective Address Mode	Register																															
EORI	8, 16, 32 CCR X N Z V C - * * 0 0	Exclusive OR Immediate	The immediate data and the destination data is exclusively ORed together and the result stored in the destination. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 <table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>Size</td><td>Effective Address Mode</td><td>Register</td></tr><tr><td colspan="8">Word data (16 bits)</td><td colspan="4">Byte data (8 bits)</td></tr><tr><td colspan="12">Long data (32 bits)</td></tr></table> Size field 00 - Byte The data is in the lower order byte of the immediate word. 01 - Word The data is the entire immediate word. 10 - Long word The data is contained in the next two immediate words. The effective address specifies the destination of the result of the operation and the following addressing modes cannot be used: 2, 10, 11, 12, 13, 14.*	0	0	0	0	1	0	1	0	Size	Effective Address Mode	Register	Word data (16 bits)								Byte data (8 bits)				Long data (32 bits)											
0	0	0	0	1	0	1	0	Size	Effective Address Mode	Register																												
Word data (16 bits)								Byte data (8 bits)																														
Long data (32 bits)																																						
EORI to CCR	8 CCR X N Z V C * * * * *	Exclusive OR Immediate to Condition Code Register	The immediate data is exclusively ORed with the CCR and the result stored in the CCR. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 <table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td colspan="8">0 0 0 0 0 0 0 0</td><td colspan="8">Byte Data</td></tr></table>	0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0	0 0 0 0 0 0 0 0								Byte Data										
0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0																							
0 0 0 0 0 0 0 0								Byte Data																														
NOT	8, 16, 32 CCR X N Z V C - * * 0 0	Logical Complement	The ones complement of the destination is taken and the results stored in the destination. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 <table><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>Size</td><td>Effective Address Mode</td><td>Register</td></tr></table> Size field 00 - Byte 01 - Word 10 - Long word The effective address specifies the destination and the following addressing modes cannot be used: 2, 10, 11, 12, 13, 14.*	0	1	0	0	0	1	1	0	Size	Effective Address Mode	Register																								
0	1	0	0	0	1	1	0	Size	Effective Address Mode	Register																												

(continued)

MICRObits (continued)**VisiCalc To Apple Plot**

Interface translates from VisiCalc to Apple Plot, prevents erroneous graphs, fits curves to data, and supplements VisiCalc with rank ordering and alphabetizing. Send SASE for details or \$30.00 for the copyable program.

Bill Starbuck
2100 E. Edgewood
Shorewood, WI 53211
(414) 963-9750

VisiCalc To Apple Writer

Veecee-Writer translated VisiCalc (/PF) files for Apple Writer 1. Send \$15.00 for the copyable program.

Bill Starbuck
2100 E. Edgewood
Shorewood, WI 53211
(414) 963-9750

TRS-80 Color Computer

Expand your 4K system to 16K for \$29.95. Expand 4K or 16K to a 32K system for only \$99. Obtain better color graphics. Full instruction/documentation provided in each kit. Two- to three-week delivery time. \$3 postage/handling charge.

Dick Williams
Computer Shed
Lane 2-1
Derry, NH 03038
(603) 432-3634

Unique VIC-20 User Group

Borrow any program from our extensive loan library for only 10% of cost and get free newsletter and special purchase prices on all VIC-20 hardware and software from our huge catalog. Membership only \$25 by check, VISA, MasterCard.

Software To Go
Rt. 3, Box 309 A 52
Clinton, TN 37716
(615) 457-5068
(615) 584-0022

68000 Software

For Apple-compatible boards (DTACK). *The Moose*: professional and only available 68000 chess program — \$67. *MUXA68*: UCSD 68000 Crossassembler — \$70. *68TICID*: Debugger — \$47. *PCON68*: UCSD-Interface for DTACK board — \$30. \$10 shipping and handling.

Moose Systems
Steenbargkoppel 21
D-2000 Hamburg 65
Germany

(Continued)

Table 1 (continued)

Mnemonic	Data Size/CCR	Function	Comments														
OR	8, 16, 32 CCR X N Z V C - . . 0 0	Inclusive OR Logical	The inclusive OR operation performs the OR operation on the source data and the destination data. The result is left in the destination.														
Opword Format																	
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																	
<table><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Register</td><td>Op Mode</td><td>Effective Address</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td>Mode</td><td>Register</td></tr></table>				1	0	0	0	Register	Op Mode	Effective Address						Mode	Register
1	0	0	0	Register	Op Mode	Effective Address											
					Mode	Register											
Register field specifies any of the 8 data registers.																	
Op Mode field																	
000 - Byte																	
001 - Word																	
010 - Long word																	
The result is stored in the specified data register.																	
The effective address specifies the source and the following addressing modes cannot be used: 2, 13, 14.*																	
Op Mode field																	
100 - Byte																	
101 - Word																	
110 - Long word																	
The result is stored in the effective address and the following addressing modes cannot be used: 1, 2, 13, 14.*																	

ORI	8, 16, 32 CCR X N Z V C - . . 0 0	Inclusive OR Immediate	The immediate data is inclusive ORed with the data in the destination and the result is left in the destination.																																								
Opword Format																																											
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																																											
<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Size</td><td>Effective Address</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Mode</td><td>Register</td></tr><tr><td colspan="4">Word data (16 bits)</td><td colspan="4">Byte data (8 bits)</td><td colspan="2"></td></tr><tr><td colspan="10">Long data (32 bits)</td></tr></table>				0	0	0	0	0	0	0	0	Size	Effective Address									Mode	Register	Word data (16 bits)				Byte data (8 bits)						Long data (32 bits)									
0	0	0	0	0	0	0	0	Size	Effective Address																																		
								Mode	Register																																		
Word data (16 bits)				Byte data (8 bits)																																							
Long data (32 bits)																																											
Size field																																											
00 - Byte The data is the lower byte of the data word.																																											
01 - Word The data is the entire 16 bits of the data word.																																											
10 - Long word The data is the two immediate words.																																											
The effective address is the destination and the following addressing modes cannot be used: 2, 10, 11, 12, 13, 14.*																																											

ORI to CCR	8 CCR X N Z V C * *	Inclusive OR Immediate data to Condition Code Register	The immediate data is inclusive ORed with the CCR and the result left in the CCR.																																
Opword Format																																			
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																																			
<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td colspan="10">Byte Data (8 bits)</td><td colspan="6"></td></tr></table>				0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	Byte Data (8 bits)															
0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0																				
Byte Data (8 bits)																																			

*The addressing modes will be covered in future issues.

MICRO

MICRObits (continued)

OSI Peter Packer

Pack wigits into boxes and ship them out in the elevators before your defective robot assistant *unpacks* them or packs you! An original arcade game of cunning and skill that increases in difficulty each round. 8K tape \$14.95.

Watts Ware
153 Madrona Drive
Anacortes, WA 98221

OSI - Affordable DBMgr

8" single/dual floppy under OS65D V3.3 video. User-defined files with formatted screen viewing and inputting. Features: find, update, delete, paging, coding, and 'screen,' 'quick,' and 'format' dump. \$55.00. Label print option — \$25.00. Report Generator (January 1983), manual only — \$10.00.

Bunin & Ward Computer Services
P.O. Box 895 Church Street Sta.
New York, NY 10008
(212) 434-5760

Low-Cost Software

Unique programs and hardware kits to adapt small computers to the real world. Control machines, make music, build test equipment and security systems, etc. For information, write to us describing your system and interests. Include stamped self-addressed envelope.

S.W. Associates
45 Furman Drive
Wayne, NJ 07470

OSI Super Defender

Play this great arcade game at home. All machine code includes: scanner, smart bombs, laser fire, moving mountains, and more. Save your humanoids from the alien landers. Very smooth (half-character moves) graphics. \$14.95 for C1, 2, 4 tape or 5 1/4" disk.

DMP Systems
319 Hampton Blvd.
Rochester, N.Y. 14612

Dynamite PET/CBM Accessories!

Write-protect switches/indicators for 2040/4040 disk drives. Real world software at low cost. 2114 RAM adapter (replaces obsolete 6550's) and 4K memory expansion for "old" 8K PETs. Hundreds of satisfied customers. Write for *free* catalog!

Optimized Data Systems
Dept. M, Box 595
Placentia, CA 92670

MICRO

Programmable Character Generator for OSI

by Colin Macauley

Design your own character set and save the characters in a form suitable for incorporation into an EPROM.

Character Generator
requires:
OSI Superboard

While developing software for a minimum chip homebrew 6502 system, it was necessary to produce a character generator. I wrote the program for an 8K OSI Superboard II to draw characters on the OSI video and save these characters in RAM. The characters could then be incorporated in an EPROM, or transferred to the homebrew system. The program was made fairly general, as the homebrew computer included the capability of a variable character depth, whereas the OSI is restricted to 8×8 characters. Although the program was intended for a specific purpose, it is equally useful in developing alternate character generators for an OSI. Thus, if games are a major attraction you may wish to define new characters [e.g., Space Invader aliens] for unused characters in your OSI character set. Accordingly, the new character set may then be loaded into a 2K EPROM (2716) and replace the original OSI character-generator ROM.

The MEMORY SIZE? cold start prompt should be restricted to 6000. This will prevent overwriting the character-generator RAM that commences at \$1800 (6144 decimal), allowing the number of characters to be 256 with a character depth of 8. The required character number is input and a display will appear on the screen to assist in the graphing of the intended character. A cursor in the top left-hand corner indicates the bit currently being altered.

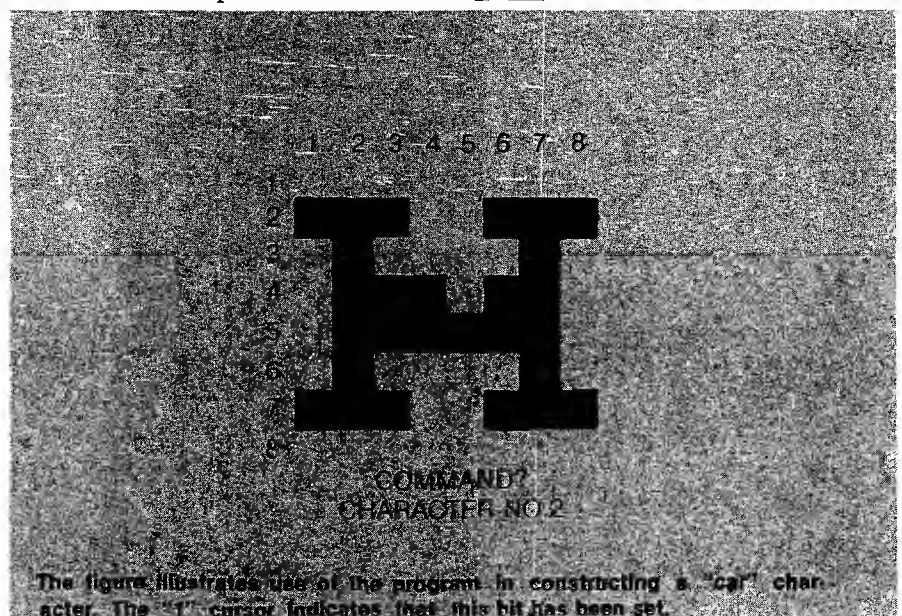
The key commands available for manipulating the cursor are as follows:

- "1" The indicated bit is set and the cursor is shifted. A block character will be inserted at the former cursor position.
- "0" The indicated bit is cleared and the cursor is shifted. A blank character will be inserted at the former cursor position.
- "H" The cursor will move from its present position to its home position (i.e., top left-hand corner of display).
- "D" The cursor will move down a row of the display.
- "F" The cursor will be shifted to the next bit without modifying the status of the previous bit.
- "ESC" Return to BASIC.
- "CR" Enter displayed character into "character-generator" RAM at nominated position.

"R" A prompt for the number of a predefined character will be requested. This character will then be displayed and may be modified to form the basis of a new character.

Set bits will be indicated by a block and cleared bits will be blanked to allow for an enlarged graphical representation of the character being created. The cursor will be either a "1" or a "0" to enable the condition of that bit to be readily identified. The 2K character generator may be saved on cassette, using well-known machine code save programs, or used directly by an EPROM programmer.

Colin Macauley is a member of the firm of Callinan and Associates, Patent Attorneys and a physicist. He uses a modified OSI Superboard II and is interested in utility-type programming. He may be contacted at 39 Shoalhaven St., Werribee, Victoria 3030, Australia.



Listing 1: Programmable Character Generator

```

4 REM LOAD USR ROUTINE
5 GOSUB350
10 FORX=1TO32:PRINT:NEXTX
20 PRINT"PROGRAMMABLE CHARACTER GENERATOR":PRINT
30 PRINT"COPYRIGHT 1981 COLIN MACAULEY":PRINT
40 INPUT"NO. OF CHARACTERS, IN GROUPS OF 16":A
50 IF(A/16)-INT(A/16)>0ORA>256THEN40
55 POKE11,162:POKE12,2
60 PRINT:INPUT"CHARACTER DEPTH. 1 TO 16":B
70 IFB>16THEN60
80 PRINT:INPUT"NEW CHARACTER SET (Y/N)":A$
90 IFMID$(A$,1,1)<>"Y"THEN110
95 REM BLANK CHAR. GEN. RAM
100 FORX=6144TO8191:POKEX,32:NEXTX
110 C=6143
120 PRINT:INPUT"CHARACTER NO.":D
130 IFD>ATHEN120
135 REM SET UP SCREEN
140 GOSUB600
210 REM USR ROUTINE SAVES REGISTERS & GETS CHAR. FROM KEYBD
220 Z=USR(Z):M=0
230 W=PEEK(216)
235 REM CHECK WHICH KEY PRESSED
236 REM "0" KEY?
240 IFW<>48THEN260
245 Q=32:GOSUB400:GOTO220
250 REM "1" KEY?
260 IFW<>49THEN270
265 Q=161:GOSUB400:GOTO220
268 REM "H" KEY?
270 IFW<>72THEN280
274 POKEV,UC:Y=53448:UC=PEEK(Y):L=1:V=Y:E=48
275 IFUC=161THENE=49
276 POKEY,E:Y=53415:GOTO220
278 REM "D" KEY?
280 IFW<>68THEN290
285 GOSUB500:GOTO220
288 REM "F" KEY?
290 IFW<>70THEN300
295 Q=UC:GOSUB400:GOTO220
298 REM "ESC" KEY?
300 IFW=27THENE=0
305 REM "CR" KEY?
310 IFW=13THEN320
315 GOSUB700:GOTO130
318 REM "R" KEY?
320 IFW=82THENGOSUB900
330 GOTO220
340 REM LOAD USR SUBR.
350 X=674:FORY=0TO15:READA:POKEY+Y,A:NEXTY
360 DATA72,138,72,152,72,32,186,255,133,216,104,168,104,170,104,96
370 RETURN
390 REM SUBR. FOR KEYS "0,1 OR F"
395 REM SHIFTS CURSOR & SETS DR RESETS INDICATED BITS
400 X=Y+(L*32)+8:P=V+1:IFP>XTHENM=L+1
410 POKEV,Q:IFM>8THEN480
420 IFM>0ANDM<>LTHEN440
430 V=P:GOTO450
440 V=Y+1+(M*32):L=M
450 UC=PEEK(V):E=48
460 IFUC=161THENE=49
470 GOTO490
480 UC=PEEK(V):E=48:IFUC=161ORUC=49THENE=49
485 IFUC=48THENU=32
490 POKEV,E:RETURN
495 REM SUBR. FOR "D" KEY-SHIFTS CURSOR DOWN A LINE
500 L=L+1:IFL>8THENL=L-1:GOTO540
510 POKEV,UC:V=V+32:UC=PEEK(V):E=48
520 IFUC=161THENE=49
530 POKEV,E
540 RETURN
590 SUBR. FOR DRAWING WORKSHEET FOR CHAR.
600 FORX=1TO32:PRINT:NEXTX
610 X=53415:F=48
620 FORZ=1TO8:POKEY+Z,F+Z:NEXTZ
640 FORZ=1TO8:W=Z:IFW>8THENW=W-10
645 POKEY+(32*Z),48+W:NEXTZ
650 Y=53448:UC=PEEK(Y):L=1:V=Y:E=48
660 IFUC=161THENE=49

```

Listing 1 (continued)

```

670 POKEY,E:Y=Y-33
680 A$="COMMAND?"
685 PRINTCHR$(13)" CHARACTER NO.":D:
690 FORX=1TO8:POKEYS4053+X,ASC(MID$(A$,X,1)):NEXTX:RETURN
695 REM SUBR. FOR "CR" KEY
698 REM SAVES CHAR. IN "CHAR. GEN." RAM AT CORRECT POSITION
700 POKEV,UC
710 Z=Y
720 FORX=1TO8
730 F=Z+(32*X):G=0
740 FORH=1TO8
750 I=PEEK(F+H):J=0:IFI=161THENJ=1
760 G=G+J:IFH=8THEN780
770 G=2*G
780 NEXTH
790 POKEC+((X-1)*A)+D,G
800 NEXTX
805 PRINT
810 INPUT"NEXT CHARACTER NO.":D
820 RETURN
880 REM SUBR. FOR "R" KEY-DRAWS REQUIRED CHAR. ON SCREEN
900 PRINT:INPUT"NO. OF CHARACTER TO BE REVIEWED":K
910 IFK>ATHEN900
920 GOSUB600:Z=Y
930 FORX=1TO8
940 F=C+((X-1)*A)+K:I=PEEK(F)
950 FORH=1TO8:R=INT(2*(H-1)+.5):N=128/R
960 J=INT(I/N)
970 IFJ=1THENPOKE(Z+(X*32)+H),161:I=I-N
980 NEXTH:NEXTX
990 UC=PEEK(Y+33):L=1:V=Y+33
1000 E=48:IFUC=161THENE=49
1010 POKEV,E
1015 IFUC=48THENU=32
1020 RETURN

```

MICRO

CSE means OSI

Software and Hardware
Specializing in C1P and C4P machines

Basic Load/SAVE:

Employs token loader system. 50-100% faster than the old indirect ASCII system. Maintains a listing of file names found on the tape

C1P.....	\$10.95
C4P.....	\$19.95*

Basic Enhancer:

Renumber, Auto Sequencer, Screen Control functions, and tape I/O system that is faster and has file names

C1P.....	\$21.95
C4P.....	\$29.95*

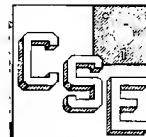
*comes with required modified monitor Rom chip

NEW! NEW! NEW!

ANCHOR SIGNALMAN MODEMS\$95.00

Please write for more info on new disk programs or send \$2.00 for catalog. Please include \$2.00 shipping (\$4.00 for modems).

**Computer
Science
Engineering**



Box 50 • 291 Huntington Ave. Boston 02115

KIDS & THE VIC

KIDS & THE APPLE

HOW TO WRITE A TRS-80

HOW TO WRITE AN IBM-PC

HOW TO WRITE AN APPLE



KIDS & THE ATARI

HOW TO WRITE A TRS-80

HOW TO WRITE AN APPLE

HOW TO WRITE AN IBM-PC

3 exceptional books join the DATAMOST library.

Here is a series of easy to read, easy to use, easy to understand books, which teach you how to write usable, useful programs on your computer. And you don't have to worry about irrelevant material which has no interest for you, because there are three specific volumes. One for the Apple,* one for the IBM-PC,* and one for the TRS-80.*

In each of these books author Ed Faulk leads you through your favorite computer and takes the mystery out of writing programs for it. As you proceed, interesting chapter by interesting chapter, you'll

wonder why you were ever intimidated by the thought of programming!

If you want to get the very most out of your Apple, IBM-PC or TRS-80 then you really want HOW TO WRITE A PROGRAM. Before you're past Chapter 2 you'll be programming. By the end of the book you'll be willing to tackle business programs, personal use programs and even games and adventures! **\$14.95**

Get your copy now. Available at computer and book stores, or:

DATAMOST (213) 709-1202
9748 Cozycroft Ave., Chatsworth, CA 91311

*Apple is a trademark of Apple Computer, Inc. IBM-PC is a trademark of IBM Corp. TRS-80 is a trademark of Tandy Corp.
VISA MASTERCHARGE accepted. \$2.00 shipping handling charge. (California residents add 6% sales tax.)

Reston Publishing Company
A Prentice-Hall Company
Reston, Virginia
Toll free (800) 336-0338

Updates and Microbes

Updates

John Beckett of Collegedale, TN, sent in this revision to "A Homespun 32K Color Computer" (53:91).

Solder the chips together rather than expecting hand-bent pins to make good contact. It is best to put a ferrite bead around the wire connected to the 6883 chip, just before it reaches the 6883. Failing this, use a 33-ohm resistor. This is done in Tandy's 32K version and is recommended by Motorola in their 6883 data sheet. Later models of the PC board have a place on the PC board where you may connect the lead from the extra bunk of chips, that avoids soldering directly to the 6883.

Myron Pulier, M.D., from Teaneck, NJ, sent in this update:

The LISZT program in the May, 1982 issue of MICRO (48:37) makes readable BASIC listings. The authors used a disk zap utility program to get lower-case characters in the DATA statements. Lacking such, I used the temporary patch, shown in listing 1, appended to LISZTER.

This patch creates new DATA strings after converting all alphabetic characters to lower case except the first one in each string. These new strings are read into a TEXT file named "DF". When this file is EXECed it replaces the LISZTER DATA statements with the new ones and displays the result for confirmation. The patch itself is removed so the converted program may be SAVED.

To operate the zap bypass program, LOAD LISZTER, type in the enclosed statements, and save the combined program as "TEMP" in case something goes wrong. Then type "RUN 1000". If the run is successful, save the program now in memory as your new copy of LISZTER.

(Continued on page 98)

1000 ***** ZAP BYPASS FOR LISZT

```
1005 D$ = Chr$(4)
      QT$ = Chr$(162)
      BR$ = QT$ + ","
1010 Print D$"OPENDF"
      Print D$"DELETEDF"
      Print D$"OPENDF"
      Print D$"WRITEDF"
1015 Print "SAVELISZTER.PATCH"
1020 Print 87"DATA";
      A = 1
      B = 25
      Gosub 2005
1025 Print 68"DATA";
      A = 26
      B = 50
      Gosub 2005
1030 Print 89"DATA";
      A = 51
      B = 51
      Gosub 2005
1035 Print 90"DATA";
      A = 52
      B = 75
      Gosub 2005
1040 Print 91"DATA";
      A = 76
      B = 107
      Gosub 2005
1045 Print "DEL 1000,3040"
      Print "INVERSE?"QT$"DATA CONVERTED"
1050 Print "NORMAL:SPEED=180:L1ST 87-91:SPEED=255"
1055 Print D$"CLOSE"
      Print D$"EXEC DF"
1060 End
```

2000 ***** CONVERT ONE LINE

```
2005 For J = A To B
2010   Read ST$
      Print QT$;
2015   LF = 0
      L = Len(ST$)
2020   If L Then
      Gosub 3005
2025   If J = B Then
      Print QT$
2030   If J < B Then
      Print BR$;
2035 Next
2040 Return
```

3000 ***** CONVERT ONE STRING

```
3005 For I = 1 To L
3010   C$ = Mid$(ST$,I,1)
3015   If "@" < C$ And C$ < Chr$(219) Then
      C$ = Chr$(Asc(C$) + 32 * LF)
      LF = 1
3020   Print C$;
Next
Return
```

END OF LISTING

PROGRAM LENGTH = 659 BYTES, TOTAL OF 27 LINE NUMBERS

51 TOTAL NON-REM STATEMENTS, 3 TOTAL REMARKS

END

...ing like it before. Nothing else like it now

... brings you continuous Hi-Res action-animation in every adventurous moment! And, real running, leaping, crawling. Real fighting, shooting, stabbing, dynamiting. Real wounding, poisoning, killing. Real action, excitement, mystery! All in a real-time challenging adventure that's the wave of the future!

Paul Stevenson's graphic genius, first displayed in the best selling "Swashbuckler" sword fighting game, outdoes itself in AZTEC. You're inside an ancient Aztec pyramid searching for the golden idol. Descend deep into the heart of the temple, meet cobras, scorpions, giant lizards, hostile Aztec guardians and more. Watch for hidden trapdoors and strange dead rooms. Be ready to fight, or run, crawl, jump to possible safety. The menacing, real, the options and strategy are yours. You've never seen an adventure game like Aztec! You'll never tire of its amazing action-animation and exciting challenges. \$39.95 for the Apple II*. At your favorite computer store.

DATA MOST

9748 Cozycroft Ave., Chatsworth, Ca 91311. (213) 700-1111

AZTEC

VISA/MASTERCARD accepted. \$2.00 handling charge. (California residents add sales tax)

*Apple II is a trademark of Apple Computer, Inc.

Utilizing the 6502's Undefined Operation Codes

by Curt Nelson, Richard Villarreal, and Rod Heisler

This method allows you to use the 6502's undefined op codes to design new and individualized pseudo-instructions under program control. A simple hardware device attached to the data bus forces a simulated BRK command when an illegal op code is detected.

Utilizing Undefined Op Codes

requires:

Hardware modification to a 6502 microcomputer

Fetch Cycle

Before the Central Processing Unit (CPU) can execute an instruction it must first get the hexadecimal code from memory. This process is called a fetch cycle. The fetch cycle is identical to the data read cycle except for the SYNC line operation, which rises to a logic level one (5V) shortly after the fetch cycle is initiated.

The fetch cycle (figure 1) starts when the system clock, ϕ_2 , falls to a logic level 0 (0V). For a 1MHz system clock the fetch cycle normally requires 1000 nano seconds, or one micro second. During this 1000 nano-second period several events occur in well-ordered sequence. First, the CPU outputs the current value of the program counter on the address bus. This is the address location of the next instruction. The specified memory then outputs the op code to the data bus. The CPU reads the op code from the data bus just before the end of the cycle.

The interval in which the Trapper has to operate extends from the time the memory device presents the op code to the data bus until the CPU latches it internally. In this time it must determine if the op code is valid or not, and force a BRK (00) if it is illegal. The Trapper described in the next section requires a maximum of 150 nano seconds to operate, leaving a mini-

mum of 525 nano seconds for the memory to present valid data to the data bus. This, of course, precludes the use of very slow memory devices but is adequate for most microcomputer systems.

Hardware

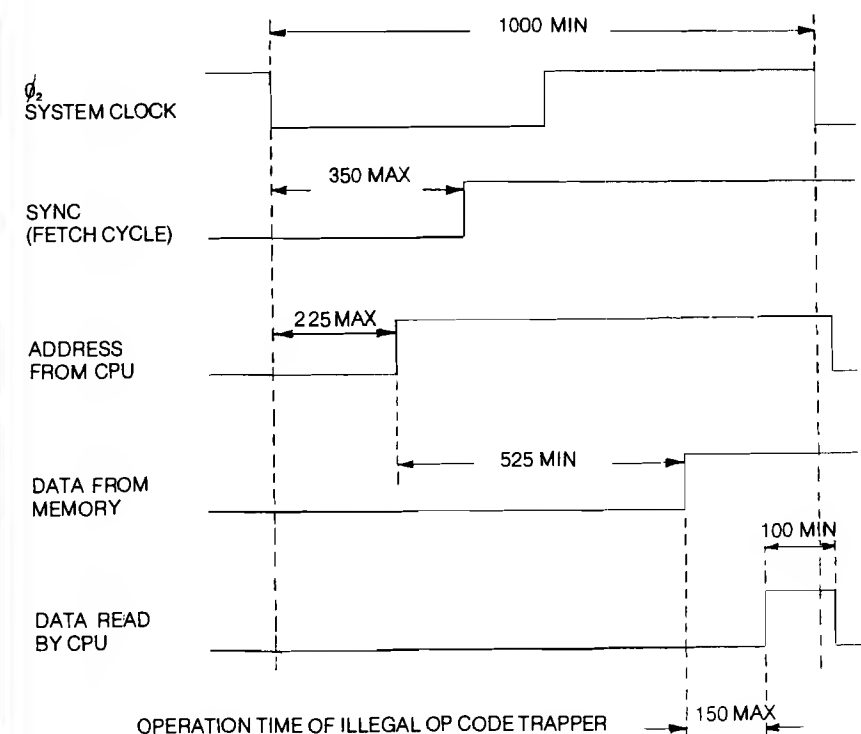
The Trapper (figure 2) samples the data bus in a parallel mode. The data lines are first buffered through IC4 and IC5 and then used to form the address to IC3, a 256×4 PROM. IC3 is always enabled and is programmed to output a logic state one for an illegal op code and a logic state zero for a legal code. Only one of the three PROM outputs is used; the others are not programmed.

The falling edge of the ϕ_2 clock in-

itiates the timing cycle for IC1, a monostable multivibrator. The output of IC1 goes high after a period of time determined by the RC network. The time-out is set for approximately 750 nano seconds. The leading edge time out from IC1 is used to clock IC2, a dual D flip-flop. The SYNC line is tied to the clear input of IC2 through two buffers. This combination of inputs to IC2 assures that its output will go high only if these three conditions are met: the SYNC line is high (fetch cycle), an illegal op code has been fetched, and IC1 has timed out.

The outputs of IC2 are used to drive open collector inverters tied directly to the data bus. When the inputs to the in-

Figure 1: Timing Diagram for the 6502 Fetch Cycle
(All times in nano (10^{-9}) seconds)



verters are high (illegal op code), the outputs force the data lines to a logic state zero, simulating a BRK command. When the inputs to the inverters are low, as under non-trapping conditions, the output appears as a high impedance to the data bus. If the data lines are pulled low, they are released when the SYNC line goes low during the next clock cycle.

Software

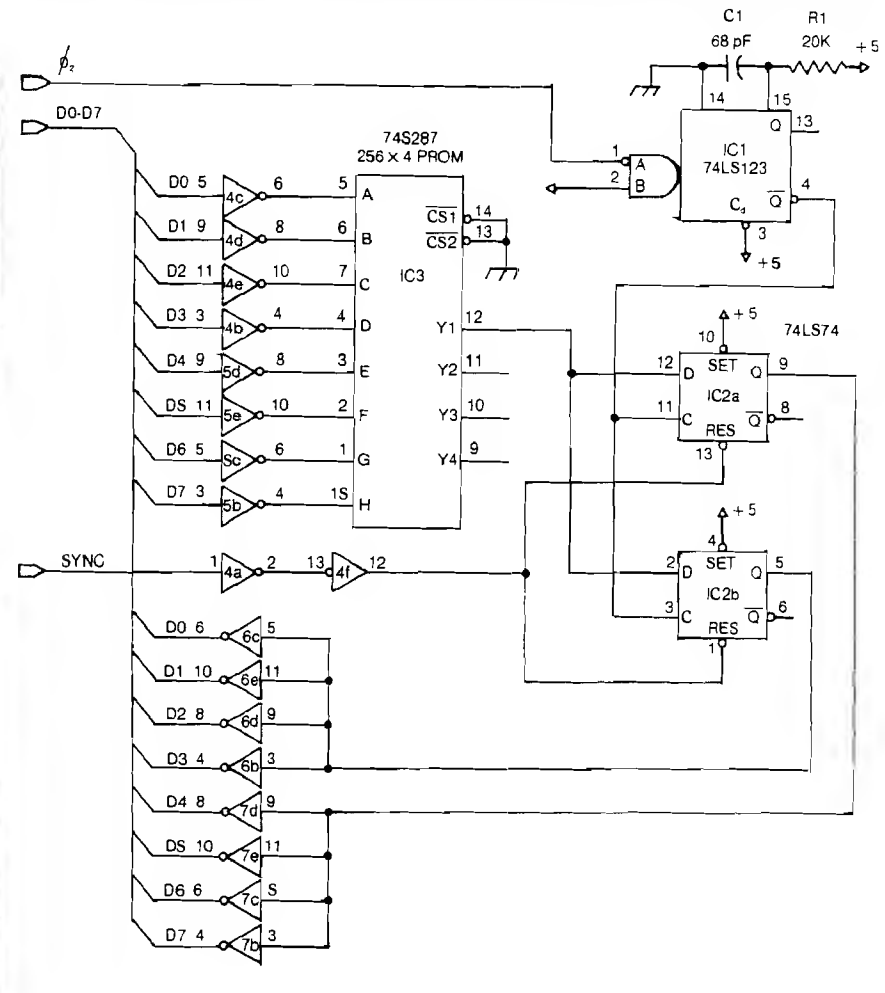
The task of the software is two-fold. First, it must determine if the break was the result of an illegal op code or a BRK instruction. Second, if the Trapper forced the break, it must retrieve the illegal op code and direct the CPU to the proper software routines.

The CPU handles the software BRK and an IRQ (Interrupt ReQuest) similarly, except for one small feature. A BRK command sets the break bit (bit four) in the processor status register. The CPU will then do an indirect jump through the IRQ vector at FFFE and FFFF. The user must load the address of the break-handling routine into the IRQ vector prior to the detection of an illegal op code, to direct the CPU to the user routine. Listing 1 shows the software used to change the IRQ vector. A starting address of \$0300 was used for the break service routine, but this is arbitrary.

The user's break-handling routine must determine whether a BRK or an IRQ was encountered. This is done by retrieving the processor status from the stack (it was automatically pushed there when the break occurred) and examining the break bit. If it is determined that bit four is set and hence a break has occurred, it retrieves the last op code. This is easily done because the address of this instruction plus two was also pushed on the stack when the program was interrupted. If this instruction was a BRK, control is passed back to the system monitor. If, on the other hand, it was an illegal op code, control is passed to a user program that implements new micro-coded instructions.

There are several methods to jump to the user code corresponding to each new instruction. The most straightforward way is to use a CMP instruction followed by a BEQ for each element in a list of new hex op codes. If more than just a few instructions are added, a more elaborate scheme may be necessary to reduce the execution time and program length. In this situation

Figure 2: Schematic diagram of the illegal op code Trapper. The board is compatible with any 6502 system bus. All lines to the board are generated by the 6502 CPU. C1 is a silver mica capacitor and R1 is a low-temperature coefficient, precision resistor.



you may want to use a jump table to build this case/select structure.

The break service routine in listing 2 is completely transparent [i.e., all registers are preserved]. The illegal op code is returned at address \$0042. The address is arbitrary and can be changed to any convenient location.

If the user exits the break service routine at line 23, indicating an IRQ, he should use the following sequence to restore the original registers:

PLA
TAX
PLP
PLA

If the routine is exited at line 40, indicating a normal BRK command, the following sequence should be used:

PLP
PLA

Programming the PROM is understood by examining figure 2. Since the system data bus is connected to the address lines of the PROM, the hex op

codes become the address to this device. Therefore, all legal op code-based addresses store 0000 and all illegal addresses store 0001.

Conclusion

This method of detecting illegal op codes is really a hardware implementation of a macro assembler directive. Although the execution time and memory space required are more than the standard JSR technique, writing and debugging programs is more straightforward when microcoded routines are

Figure 3

Number	Type	+5V	Gnd
IC1	74LS123	16	8
IC2	74LS74	14	7
IC3	74S287	16	8
IC4,5	74LS04	14	7
IC6,7	7405	14	7

Listing 1: Software to modify the IRQ vector to point to a user program.

```

0800      1  ;SETTING UP THE IRQ VECTOR
0800      2  ;
0800      3  ;
0200      4      ORG $200
0300      5  USRPRG  EQU $0300      ;ADDRESS OF USER PROGRAM
FFFF      6  IRQLOW  EQU $FFFF     ;LOW ADDRESS OF IRQ VECTOR
FFFF      7  IRQHIG  EQU IRQLOW+$1 ;HIGH ADDRESS OF IRQ VECTOR
0200      8  ;
0200      9  ;
0200     10  ;INITIALIZATION
0200     11  ;
0200     12  ;
0200 A9 00   13      LDA #USRPRG      ;SET IRQ VECTOR TO USER BREAK
                                ROUTINE
0202 8D FE FF 14      STA IRQLOW
0205 A9 03   15      LDA /USRPRG
0207 8D FE FF 16      STA IRQHIG
020A      17  ;
020A      18  ;
020A      19  ;
020A      20  ;
020A      21  ;MAIN PROGRAM

```

Listing 2: Program to handle a break service routine. Determines whether a break or an IRQ has interrupted the system and transfers control to the proper location.

```

0800      1  ;BREAK SERVICE ROUTINE
0800      2  ;
0800      3  ;
0800      4  ;
0300      5      ORG $300
0380      6  IRQSER  EQU $380      ;STANDARD IRQ SERVICE
03A0      7  USRBRK  EQU $3A0      ;STANDARD BREAK SERVICE
0040      8  SAVLOW  EPZ $40
0041      9  SAVHIG  EPZ SAVLOW+$1
0042     10  SAVOPC  EPZ SAVHIG+$1
0104     11  FLAG    EQU $104
0105     12  ADDLOW  EQU $105
0106     13  ADDHIG  EQU ADDLOW+$1
0300     14  ;
0300     15  ;
0300 48      16      PHA              ;PRESERVE ACC
0301 08      17      PHP              ;PRESERVE FLAGS
0302 8A      18      TXA
0303 48      19      PHA              ;PRESERVE X
0304 BA      20      TSX
0305 BD 04 01 21      LDA FLAG,X      ;GET FLAGS
0308 29 10   22      AND #$10
030A F0 74   23      BEQ IRQSER
030C BD 06 01 24      LDA ADDHIG,X    ;GET ADD + 2 FROM STACK
030F 85 41   25      STA SAVHIG
0311 BD 05 01 26      LDA ADDLOW,X
0314 85 40   27      STA SAVLOW
0316 D0 02   28      BNE SKIP          ;BR IF NOT ON PAGE BOUNDRY
0318 C6 41   29      DEC SAVHIG        ;DEC PAGE
031A C6 40   30      DEC SAVLOW        ;DEC ILLEGAL OPCODE ADDRESS
031C D0 02   31      BNE SKIP1        ;BR IF NO PAGE CROSSED
031E C6 41   32      DEC SAVHIG        ;DEC PAGE
0320 C6 40   33      DEC SAVLOW        ;DEC ADDRESS AGAIN
0322 A2 00   34      LDX #000         ;INDEX
0324 A1 40   35      LDA (SAVLOW,X)    ;GET ILLEGAL OP CODE
0326 85 42   36      STA SAVOPC       ;PRESERVE IT
0328 68      37      PLA
0329 AA      38      TAX              ;RESTORE X
032A A5 42   39      LDA SAVOPC       ;RETRIEVE ILLEGAL OP CODE
032C F0 72   40      BEQ USRBRK       ;BR FOR NORMAL BREAK
032E 28      41      PLP              ;RESTORE FLAGS
032F 68      42      PLA              ;RESTORE ACC
0330      43  ;
0330      44  ;
0330      45  ;
0330      46  ;USER ROUTINES
0330      47  ;
0330      48  ;
0330      49  ;
0330     50  ;RETURN TO MAIN PROGRAM
0330     51  ;
0330     52  ;
0330 E6 40   53      INC SAVLOW        ;BUMP LOW ADDRESS
0332 D0 02   54      BNE SKIP2        ;BR IF NO PAGE CROSSED
0334 E6 41   55      INC SAVHIG       ;BUMP PAGE
0336 C6 40 00 56      SKIP2  JMP (SAVLOW)
0339      57      END

```


incorporated into your program as simple instructions.

A few words of caution: first, it is necessary to acquaint yourself with the user-available monitor subroutines on your system. The SYM-1, for example, has monitor routines to do some of the functions in listing 2. The Apple, as well, has monitor routines that can be used to shorten this program. Second, the illegal op code FF rearranges the stack and hence should be avoided.

You are now in a position to expand the instruction set of your 6502-based system. What instructions should you add? Here are a few suggestions: integer multiply and divide, double precision math operations, jump indirect-indexed, push and pull to a user stack, and memory to memory transfer. You can even add a pseudo B accumulator and a 16-bit index register.

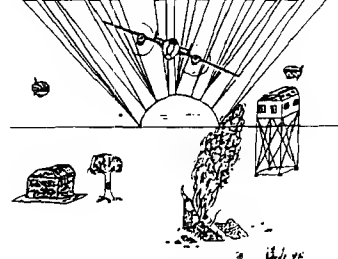
The authors may be contacted at the School of Engineering, Walla Walla College, College Place, Washington 99324.

MICRO™

Interesting Software 
presents

OSI C4P-MF SOFTWARE

AIR PATROL



YOU MUST PILOT YOUR WWII VINTAGE AIRCRAFT ACROSS A SCROLLING LANDSCAPE AND RESCUE POW'S IN ENEMY TERRITORY. SOME OF THE SMOOTHEST GRAPHICS EVER SEEN ON AN OSI! IT ALSO USES A NEW TECHNIQUE OF USING "LARGE" MULTI-CHARACTER SHAPES FOR A REALISTIC GAME. YOU WILL REALLY LOVE THIS ONE! PLEASE SPECIFY WHETHER YOU WANT JOYSTICK OR KEYBOARD OPTIONS. THIS GAME IS SO EXTENSIVE THAT IT TAKES UP THE ENTIRE DISK!

ALL THIS FOR ONLY \$19.95

SEND TO:
INTERESTING SOFTWARE
21101 S. HARVARD BLVD.
TORRANCE, CA 90501
(213) 328-9422

Calif. residents
add sales tax

SIG-FORTH V 1.0

The only stand-alone Forth system
for O.S.I. serial machines

Features:

Complete Forth source code
Advanced Screen editor w/source
6502 macro assembler w/source
Double number and CASE extensions
Vectored boot capability
Several Utility Screens
Complete glossary

Dos Includes:

Bi-Directional NEC driver
65U read capability
NMHZ Capability

\$100.00

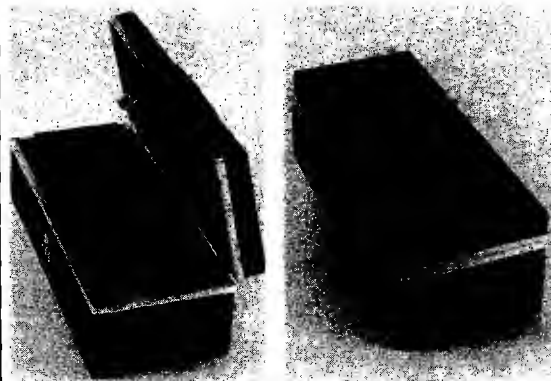
POSTAGE
PAID

DIGI COM ENGINEERING, INC.

P.O. Box 1454
Kodiak, Alaska 99615

ORDERING INFORMATION: Check, money order or C.O.D.'s accepted.
Shipment VIA first class mail. Allow approximately one week for delivery.

FRIGHTENED BY A PLASTIC CASE



Were you ever frightened to carry your 5 1/4 floppies out of the security of your home or office, because of that flimsy plastic case you keep your discs in.

Store and carry 100 discs safely and securely

Each disc pocket is surrounded by an inch of high density foam inhibits crushing, jamming, summer's heat, and winter's cold.

100 disc 5 1/4 floppy carrying case \$39.95 ea

Dealer Inquiries Invited. Phone or mail orders accepted.

Check, Money order, MasterCard, Visa

For Shipping add \$2.00. N.Y. State residents Add Sales Tax

Product Design and Copyright © 1982 By Greg Carbonaro

Unique Software Inc.

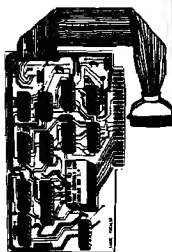
POST OFFICE BOX N, DEER PARK, NEW YORK 11729 (516) 666-7577

APPLE

HARDWARE

THE TACKLER™ — DUAL • MODE PARALLEL INTERFACE FOR THE APPLE® 2 BOARDS IN ONE FOR NO MORE COMPATIBILITY PROBLEMS!

An intelligent board to provide easy control of your printer's full potential. Plus a standard parallel board at the flip of a switch — your assurance of compatibility with essentially all software for the APPLE®. Hires printing with simple keyboard commands that replace hard to use software routines. No disks to load. Special features include inverse, doubled, and rotated graphics and many text control features, available through easy keyboard or software commands. Uses industry standard graphics commands. This is the first truly universal intelligent parallel interface! Change printers — no need to buy another board. Just plug in one of our ROM'S and you're all set. ROM'S available for Epson, C. Itoh, NEC, and Okidata — others available soon. Specify printer when ordering. Call for Price.

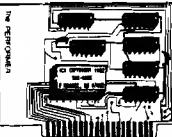


THE UPGRADEABLE PPC-100 PARALLEL PRINTER CARD

A Universal Centronics type parallel printer board complete with cable and connector. This unique board allows you to turn on and off the high bit so that you can access additional features in many printers. Easily upgradeable to a fully intelligent printer board with graphics and text dumps. Use with EPSON, C. ITOH, ANADEX, STAR-WRITER, NEC, OKI and others with standard Centronics configuration. **\$139.00**

IF YOU WANT GRAPHICS AND FORMATTING THEN CHOOSE THE PERFORMER

for Epson, OKI, NEC 8023, C. ITOH 8510 provides resident HIRES screen dump and print formatting in firmware. Plugs into Apple slot and easy access to all printer fonts through menu with PR# command. Use with standard printer cards to add intelligence. **\$49.00** specify printer



THE MIRROR FIRMWARE FOR NOVATION APPLE CAT II®

The Data Communication Handler ROM Emulates syntax of an other popular Apple Modem product with improvements. Plugs directly on Apple CAT II Board. Supports Vindex and Smarterm 80 column cards, touch tone and rotary dial, remote terminal, voice toggle, easy printer access and much more. List **\$39.00** Introductory Price **\$29.00**

MINI ROM BOARDS

Place your 2K program on our Mini Rom Board. Room for one 2716 EPROM. Use in any slot but zero. Only **\$34.95**

DOUBLE DOS Plus

A piggy-back board that plugs into the disk-controller card so that you can switch select between DOS 3.2 and DOS 3.3 DOUBLE DOS Plus requires APPLE DOS ROMS. **\$39.00**

SOFTWARE

APPLE

Super Pix

Hires screendump software for the Epson, OKI, C. Itoh and Nec 8023. Use with Tymac PPC-100. Special **\$19.95** (Specify Printer)

Mr. Lister — Customer Contact Profiler & Mailer

A Super Mail List Plus more — up to 1000 Entries on single 3.3 Disk (only 1 Drive required) — 2 second access time to any name — full sort capabilities — Dual Index Modes — supports new 9 digit Zip. Easy to follow manual — Not Copy Protected — 4 user defined tables with 26 sort selections per table — Beta tested for 6 months — user defined label generation. Introductory Price **\$135**. **\$99.00** Dealer & Dist. Inquiries Invited.

APPLE LINK

A communications system for the Apple® (Requires Hayes Micro Modem) Transmit and receive any type of file between APPLES®, Automatic multi-file transfer, real time clock indicating file transfer time. Complete error check Plus conversation mode. Only one package needed for full transfers. Compatible with all DOS file types. (requires Hayes Micro Modem) **\$59.00**

THE APPLE CARD/ATARI CARD

Two sided 100% plastic reference card Loaded with information of interest to all Apple and Atari owners. **\$3.98**

NIBBLES AWAY II

AGAIN! Ahead of all others.

- **AUTO-LOAD PARAMETERS** . . . Free's the user from having to Manually Key in Param values used with the more popular software packages available for the Apple II.
- **EXPANDED USER MANUAL** . . . incorporates new Tutorials for all levels of expertise; Beginners Flowchart for 'where do I begin' to 'Advanced Disk Analysis' is included.
- **TRACK/SECTOR EDITOR** . . . An all new Track/Sector Editor, including the following features: Read, Write, Insert, Delete Search, and impressive Print capabilities!
- **DISK DIAGNOSTICS** . . . Checks such things as: Drive Speed, Diskette Media Reliability, and Erasing Diskettes.
- **HIGHEST RATED** . . . Best back up Program in Softalk Poll (Rated 8.25 out of 10).
- **CONTINUAL UPDATES** . . . Available from Computer Applications and new listings on the source. **\$69.95**

Dealer and Distributor Inquiries Invited.



MICRO-WARE DIST. INC.
P.O. BOX 113
POMPTON PLAINS, N.J. 07444

201-838-9027

SuperPET Characters

Terry M. Peterson, 8628 Edgehill Ct.,
El Cerrito, CA 94530

The SuperPET contains a 4K character-generator ROM in place of the 2K ROM found in normal CBM 8032s. The 4K ROM contains four character sets. In addition to the two PET/CBM character sets found in the 2K ROM, there are two new sets designed by Waterloo Computing Systems — ASCII and APL. The Waterloo ASCII character set is used in the SuperPET by all the Waterloo Micro languages except MicroAPL. This article describes some of the features of the Waterloo ASCII character set that are not well-covered in the Waterloo documentation accompanying the SuperPET.

All the printable ASCII characters — codes 32 to 127 — in the Waterloo ASCII set are pure ASCII. By this I mean they are all recognizable duplicates of the corresponding character found in an ASCII table. Furthermore, the PRINTed codes are *identical* to the screen POKE codes for a given character! Many of the screen control codes are consistent with normal printer usage; e.g., cursor-down = 10 [LF], cursor-back = 8 [BS], and clear-screen = 12 [FF]. This means that turning

neatly formatted CRT output into neatly formatted hardcopy on an ASCII printer (like the MX-80) is much easier than with the CBM character set (the one Gary Huckel of TNW so appropriately calls 'half-ASCII').

Notice I said the printable characters, 32 to 127, have the same PRINT and POKE codes; but what about POKEing the ASCII control codes 0 to 31? By experiment you will find these codes do not all cause the same action when POKEd as when PRINTed. The POKE characters and PRINT actions of these codes are shown in table 1. The codes 0 and 14-30 give an odd little white box when POKEd or PRINTed. Code 31 gives the Greek letter μ , POKEd or PRINTed. Codes 1-11, when POKEd, give eleven line graphic characters that are useful for drawing outline boxes or grids. These characters are similar to the graphics characters available on the Epson MX printers with Graphtrax Plus. They are also very like one subset of the CBM graphics characters; the shifted-zero is an example (see table 1). When PRINTed, most of the codes from 1 to 13 perform some sort of control function, as shown in table 1.

What about the high-order bit that gives the codes 128 to 255? Either PRINTed or POKEd, all the codes from 128 to 255 reproduce, in reverse field, their X-minus-128 POKEd counterparts. Although all these reverse-field characters are available (and Waterloo

didn't usurp the RVS key for another function), Waterloo ASCII apparently has no reverse control code such as in the CBM character set. Therefore, to print a reverse-field string, each character must be extracted from the string and transformed by adding 128. For example in microBASIC:

```
FOR I = 1 TO LEN(CHARSTRINGS)
CHAR$ = STR$(CHARSTRINGS,I,1)
RVSCHAR$ = CHR$(128 + ORD
(CHAR$))
PRINT RVSCHAR$;
NEXT I
```

Perhaps this encumbrance is the reason reverse-field characters aren't mentioned in Waterloo's documentation?

VIC Jitter Fix

David Malmberg, 43064 Via Moraga,
Fremont, CA 94539

In my October 1981 MICRO article [41:54], "VIC Light Pen-Manship," I pointed out that the locations in the VIC chip that return the light pen's horizontal screen position (\$9006) and vertical screen position (\$9007) are

Table 1

Code	Mnemonic	ASCII Name	Print Action	POKE Character	CBM Graphics Equivalent	Epson Graphtrax+ Equivalent
1	SOH	Start Heading	Home cursor	Vertical line	CHR\$(221)	CHR\$(156)
2	STX	Start TeXt	? [Run]	Horizontal line	CHR\$(195)	CHR\$(157)
3	ETX	End TeXt	? [Stop]	Lower right corner	CHR\$(189)	CHR\$(154)
4	EOT	End Transmission	Delete	Lower left corner	CHR\$(173)	CHR\$(153)
5	ENQ	ENQuiry	Insert	Upper left corner	CHR\$(176)	CHR\$(134)
6	ACK	ACKnowledge	Erase to EOL	Upper right corner	CHR\$(174)	CHR\$(149)
7	BEL	ring BELl	Cursor right(!)	Bottom middle corner	CHR\$(177)	CHR\$(158)
8	BS	Back Space	Cursor left	Left middle corner	CHR\$(171)	CHR\$(150)
9	HT	Horizontal Tab	Tab	Top middle corner	CHR\$(178)	CHR\$(152)
10	LF	Line Feed	Cursor down	Right middle corner	CHR\$(179)	CHR\$(151)
11	VT	Vertical Tab	Cursor up	Cross	CHR\$(219)	CHR\$(159)
12	FF	Form Feed	Clear screen	Little white box		
13	CR	Carriage Return	Carriage return	Little white box		

Updates and Microbes

(Continued from page 91)

Robert R. Ringel of Comstock Park, MI, found a bug in COMPRESS (52:89):

If COMPRESS is processing the token for NEXT (\$82) one byte before a page boundary, it can lose that token when it goes to update its addresses for the new page.

To correct this problem, replace the STX instruction at \$9088 with \$86E3 and the corresponding LDX instruction at \$908E with \$A6E3. Zero page location \$E3 is an unused location that works well for a temporary location in this instance.

COMPRESS Removes Variables

Warren Friedman, from Berkeley, CA, sent in this update:

The program COMPRESS, well written and clearly described by Barton M. Bauers (MICRO 52:89) removes any variable names appearing after NEXT statements. It does this by ignoring all characters until the following colon or the end of the program line (see \$93EC - \$93EF). This could cause problems in two cases.

The first problem occurs when several variables are used with one NEXT, as in NEXT I,J. The second case is when a NEXT variable *must* be stated. This may occur with nested loops in which the inner loop NEXT is the result of an IF...THEN statement. (Editor's note: A poor programming practice. Loops should be cleared before exiting or else stack overflow can occur.)

These problems with NEXT can be solved by treating NEXT in the same way an IF statement is dealt with, which is to leave it as the programmer wrote it. (Bauers calls this a Terminal Command.) This is done by changing one byte of COMPRESS. First BLOAD COMPRESS, then, in BASIC, POKE 37871,72 [or, in the monitor, enter 93EF:48]. Then BSAVE COMPRESS, A\$9000,L\$600.

Similarly, programmers who use & statements (and who do not mind having LET statements remain in the program, if there are any) can change lines 460 and 461. In BASIC, POKE 37873,202 : POKE 37874,240 : POKE 37875,68 [or, in the monitor, enter 93F1:CA F0 44]. The two lines of COMPRESS become

```
C9 CA CMP #SCA ;is it '&'?
F0 44 BEQ IF ;yes
```

MICRO™

Short Subjects (continued)

subject to noise. These noisy registers can cause the pen's readings to jitter about the screen. The October article presented a machine-language routine that eliminated this jitter problem by taking seven separate readings of the pen's coordinates, sorting them, and returning the median readings (thus ignoring the jittery readings that should be at one extreme or the other of the sorted list). This routine also calculated the light pen's screen row and column for the special case of an Atari or Commodore light pen.

Having recently experimented with the use of the Atari VCS's game paddles with the VIC, I discovered that the left (\$9008) and right (\$9009) game paddle registers also suffer from jitter problems. This can be very frustrating when you are playing a paddle game like PONG or BREAKOUT and the paddles occasionally bounce around the screen as if they were possessed by evil computer spirits. The severity of the problem seems to be a function of the game paddle unit itself — my neighbor's paddles are much noisier than mine.

The BASIC subroutine, given in listing 1, POKes into the VIC's cassette buffer a machine-language routine that provides a general solution to this jitter problem. To use the routine in your

paddle programs, follow these steps: 1. append the subroutine to your game paddle program, 2. GOSUB 1000 at the start of the program to load the machine code into the cassette buffer, 3. SYS(828) to read *both* paddle registers, and 4. get the left paddle's un-jittered reading by PEEKing 936 and the right by PEEKing 937. Be sure to use this routine cautiously in any program that is doing tape input or output because of the risk of clobbering the machine code in the cassette buffer.

This same routine may also be used to un-jitter the light pen registers by deleting lines 1190 and 1200. The resulting machine code is more universal than the version given in the October 1981 article because it can be used with any light pen, rather than just the Atari and Commodore pens.

Should other VIC chip registers be discovered that suffer from jitter, they can be easily handled with this routine by merely POKeing the low byte of their addresses into locations 835 and 857. See line 1190 of the listing where this is done for the game-paddle registers.

Because this program is very similar to the one presented in my previous article, a full assembly listing is not given.

Jitter Fixer Subroutine

```
1000 REM MACHINE LANGUAGE ROUTINE TO READ "JITTERY" VIC LOCATIONS
1010 REM SUCH AS LIGHT PEN COORDINATES OR GAME PADDLE SETTINGS
1020 REM SYS(828) TO READ --- VALUES RETURNED IN LOCATIONS 936 AND 937
1030 FOR I= 828 TO 935 :READ DC:POKE I,DC:NEXT I
1040 DATA 162,0,160,3,132,152,173,6,144
1050 DATA 160,171,132,151,32,133,3,165
1060 DATA 151,24,109,170,3,133,151,144,2
1070 DATA 230,152,173,7,144,32,133,3,232,236
1080 DATA 170,3,240,9,165,162,197,162,240
1090 DATA 252,76,62,3,173,170,3,74,168
1100 DATA 177,151,141,169,3,169,171,133
1110 DATA 151,169,3,133,152,177,151,141
1120 DATA 169,3,96,142,168,3,172,168,3
1130 DATA 192,0,240,22,136,209,151,200
1140 DATA 176,16,136,141,168,3,177,151
1150 DATA 200,145,151,136,173,168,3,56
1160 DATA 176,230,145,151,96,0,0,7
1170 REM ROUTINE WILL NORMALLY READ GAME PADDLES
1180 REM TO READ LIGHT PEN COORDINATES, DELETE THE NEXT TWO STATEMENTS
1190 POKE 835,3:POKE 857,9
1200 POKE 868,169:POKE 869,255:POKE 870,233:POKE 871,1:POKE 872,208
1210 RETURN
```

MICRO™

MICRO™

New Publications

So we can list more of the many new books now available, we are offering New Publications in a different format. We think you'll find this increased sampling of computer literature useful.

Library of PET Subroutines, by Nick Hampshire. Hayden Book Company, Inc. (Rochelle Park, NJ), 1982, 140 pages, paperback.

ISBN: 0-8104-1050-8 \$14.95

PET Graphics, by Nick Hampshire. Hayden Book Co., Inc. (Rochelle Park, NJ), 1982, 218 pages, paperback.

ISBN: 0-8104-1051-6 \$16.95

Computer Consciousness: Surviving the Automated 80's, by H. Dominic Covey and Neil Harding McAlister, Addison-Wesley Publishing Company, Inc. (Reading, MA), 1982, 211 pages, paperback.

ISBN: 0-201-01939-6 \$6.95

Atari Sound and Graphics, by Herb Moore, Judy Lower, and Bob Albrecht. John Wiley & Sons, Inc. (605 Third Ave., N.Y.C., NY 10158), 1982, 234 pages, paperback.

ISBN: 0-471-09593-1 \$9.95

The Creative Apple, Edited by Mark Pelczarski and Joe Tate. Creative Computing Press (Morris Plains, NJ), 1982, 448 pages, paperback.

ISBN: 0-916688-25-9 \$15.95

The VisiCalc Book, Apple Edition, by Donald H. Beil, Reston Publishing Company, Inc. (Reston, VA), 1982, 301 pages, paperback.

ISBN: 0-8359-8398-6 \$14.95

The Third Book of Ohio Scientific, by S. Roberts. ELCOMP Publishing, Inc. (Postbox 1194, Pomona, CA 91769), 1982, 137 pages, 5¼ × 8¼ inches, paperback.

ISBN: 3-921682-77-0 \$17.95

Kilobaud Klassroom, by George Young and Peter Stark. Wayne Green Books (Peterborough, NH 03458), 1982, 419 pages, 6 × 9 inches, paperback.

ISBN: 0-88006-027-1 \$14.95

Computers for Kids, by Sally Greenwood Larson. Creative Computing Press (P.O. Box 789-M, Morristown, NJ 07960), 1981, 73 pages, paperback.

ISBN: 0-916688-21-6 \$4.95

Ciarcia's Circuit Cellar, Volume III, by Steve Ciarcia. BYTE/McGraw-Hill (70 Main St., Peterborough, NH 03458), 1982, 228 pages, 8¼ × 11 inches, paperback.

ISBN: 0-07-010965-6 \$12.95

Techniques for Creating Golden Delicious Games for the Apple Computer, by Howard M. Franklin, Joanne Koltnow, and Leroy Finkel. John Wiley and Sons, Inc. (605 Third Ave., N.Y.C., NY 10158), 1982, 150 pages, paperback.

ISBN: 0-471-09083-2 \$12.95

BASIC for Business by Douglas Hergert. SYBEX (2344 Sixth Street, Berkeley, CA 94710), 1982, 223 pages, 7 × 9 inches, paperback.

ISBN: 0-89588-080-6 \$12.95

Computers for People by Jerry Willis and Merl Miller. Dilithium Press (P.O. Box 606, Beaverton, OR 97075), 1982, 200 pages, 5¼ × 8½ inches, paperback.

ISBN: 0-918398-64-9 \$7.95

MICRO™

BUY! SELL! TRADE!
COMPUTER & HAM EQUIPMENT

**COMPUTER®
TRADER**

**PERMANENT
SUBSCRIPTION
\$10.00**

Low Ad Rates — Mailed Monthly

FOREIGN SUBSCRIPTIONS \$25.00 YEAR

COMPUTER TRADER®

Chet Lambert, W4WDR

1704 Sam Drive • Birmingham, AL 35235
(205) 854-0271

Please include your Name, Address, Call Sign or Phone Number

QCB-9 SINGLE BOARD COMPUTER

- 6809 BASED
- RUNS TSC FLEX DOS
 - ★ QCB-9/1 S-100 BUS
 - ★ QCB-9/2 SS-50 BUS

\$149.00*
*PARTIAL KIT

FEATURES

- 5¼" Floppy Controller
- Serial RS-232 Port
- Centronics Type Printer Port
- Keyboard/Parallel Port
- 24K Bytes of Memory
- QBUG Resident Monitor
- 6802 Adaptor

**FULLY ASSEMBLED
& TESTED \$389.00**

- 48-hour Burn-in
- 90 Day Warranty

NAKED-09 SS-50 6809 CPU CARD \$49.95*

- ★ 1K OF RAM AT E400 Assembled & Tested \$149.00 PCB & Documentation Only
- ★ 6K OF EPROM AT E800-FFFF 2 MHz Version \$189.00
- ★ HIGH QUALITY DOUBLE SIDED PCB ★ SOLDER MASKED ★ SILK SCREENED

TSC, FLEX DOS, ASSEMBLER, EDITOR \$150.00

QBUG RESIDENT MONITOR \$50.00

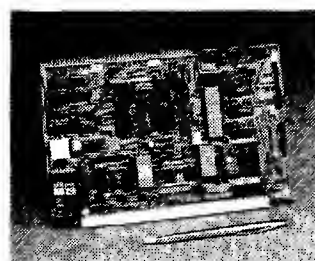
- ★ Disc Boot
- ★ Memory Test
- ★ Break Points
- ★ Memory Exam & Exchange
- ★ Zero Memory
- ★ Jump to User Program
- ★ Memory Dump
- ★ Fill Memory
- ★ Register Display & Change

QBUG IS A TRADEMARK OF LOGICAL DEVICES INC. Copyright 1981

PHONE ORDERS: (305) 776-5870

LOGICAL DEVICES INC. COMPUTER PRODUCTS DIVISION

781 W. OAKLAND PARK BLVD. • FT. LAUDERDALE, FL 33311
TWX: 510-955-9496 • WE ACCEPT VISA, MC, CHECKS, C.O.D., MONEY ORDER



Reviews in Brief

Product Name: **Spellmaster**
Equip. req'd: Commodore 80-column screen and dual disk (40- and 64-column versions expected soon)
For Wordpro files (Wordcraft & Silicon Office versions planned)
Uses functional 4K ROM at \$9000
Price: \$195
\$ 75 for legal or medical dictionary options
Manufacturer: Management Systems Alternatives
6219 Thirteenth Avenue South
Gulfport, FL 33707

Description: Finally, a decent spelling checker for CBM computers! Highly recommended for word-processing writers who do not spell well.

Pluses: It is far faster than its only competitor and has an honest 40,000-word dictionary. *Spellmaster* presents suspect words for editing in context in reverse field on a typical Wordpro screen display. Suspect words may then be easily corrected or added to the dictionary for future reference (up to 3,000 more words on the 4040, and 20,000 more on the 8050). Corrected files are resaved to disk, avoiding the hassle of reloading the word processor and searching for the errors. The program is mostly self-documenting, though it comes with a typical manual. There is a HELP screen in the program and useful prompts throughout.

Minuses: When editing, it is easy to skip past a word that needs to be repaired or added to the dictionary. At present, there is no way to back up except by aborting and restarting the edit. The company is attempting a fix.

Skill level required: Users should be fairly familiar with Wordpro and willing to spend about an hour reading the *Spellmaster* manual before use.

Reviewer: Jim Strasma

Product Name: **Electric Duet**
Equip. req'd: Apple II or Apple II Plus
Price: \$29.95
Manufacturer: Insoft
10175 Barbur Blvd., Suite 202B
Portland, OR 97219
Author: Paul Lutus
Copy Protection: Yes
Language: 6502 Assembly
Description: A software-only music synthesis system for generating 2-part music on an Apple with no additional hardware required.

Pluses: An external speaker can be used to improve fidelity via the cassette port. The package includes a music editor for constructing tunes, with several sample tunes. A combined display allows for the simultaneous entering and playing of music. Entered scores can be transposed both in key and in tempo. Each note played may have one of four voices. Notes can be entered either into an editor or played directly from the keyboard. Then the music can be incorporated directly into user programs! The storage format of the music is described for the more advanced programmer who may wish to access the binary score directly.

Minuses: The manual is brief (17 pages) but complete. Although the author has permitted the user to play music directly from the Apple keyboard (using the upper row of keys for one note and the lower for the other), I personally found this feature awkward to use. The editor is much more complete for entering music from the keyboard. As mentioned in the manual it is included only for familiarization. Deletion of a line using the music editor is not a single stroke command. To accomplish a line deletion, a file must be opened so that the line to be deleted is the last. Then deletion will remove it. After working with Musicomp, Paul Lutus' first music editor, I was spoiled by his hi-res display of notes in motion. I would love to have seen that feature retained in *Electric Duet*. However, by obtaining 2-part music with no hardware, at a fraction of the cost of popular music boards, this program should be considered carefully before investing in more expensive alternatives.

Skill level required: Fairly easy for the novice to master with a little practice.

Reviewer: David Morganstein

Product Name: **Terminal-40**
Equip. req'd: VIC-20
8K (or more) of extra memory
VICMODEM or RS-232 compatible modem
Price: \$29.92
Manufacturer: Midwest Micro Associates
P.O. Box 6148
Kansas City, MO 64110
Author: Dr. Jim Rothwell

Description: *Terminal-40* is an extremely powerful telecommunications program for the VIC-20. This machine-language program is fast enough to support up to 2400 baud, is quite flexible, and allows you to specify duplex, parity, wordsize, stopbits, linefeed, and baud rate options. Through software, *Terminal-40* displays a 40-character line with each character represented by a 3 x 6 matrix. All characters are shown as upper case and are quite readable. *Terminal-40* also has a 4K or larger buffer,

Reviews in Brief (continued)

which can be used to capture copies of the material being transmitted or received for later study or dumping to the printer.

Pluses: A versatile and exceedingly well-done package. The 40-column display is great!

Minuses: Although *Terminal-40* supports the printer, it does not handle the disk, nor is there any way to use it to transmit or receive a program. The program comes on an "auto-start" tape and cannot be copied to disk or another tape.

Documentation: The 20-page manual is clear and comprehensive.

No special skills required.

Reviewer: David Malmberg

Product Name: **Doubletime Printer**
Equip. req'd: Apple II Plus
Any of the popular printers
Price: \$99.95
Manufacturer: Southwestern Data Systems
P.O. Box 582
Santee, CA 92071
(714) 562-3221

Description: *Double Printer* permits printing to take place as a background task. You can continue to use your computer while it is printing rather than being "frozen out." This should prove particularly valuable in word processing applications.

Pluses: The product is extremely versatile. Applesoft, binary, or text files are printed without conversion. Formatting commands are available and easy to use.

Minuses: The product is not easy to get up and running. It requires a ROM chip change, a board installation, and a diskette boot. All this could be dealer-performed for the more timid user. It is worth the trouble.

Documentation: The instructions are well-written but quite technical.

Skill level required: An intermediate familiarity with the Apple is necessary.


Reviewer: Chris Williams

Product Name: **Apple-Cillin II**
Equip. req'd: Apple II or Apple II Plus with disk drive (13- or 16-sector)
Price: \$49.95
Manufacturer: XPS, Inc.
323 York Road
Carlisle, PA 17013

Description: This diagnostic utility tests RAM and ROM chips, the disk system, peripheral cards, keyboard, CRT display, printer, tape recorder, game controls, and CPU

(Continued on next page)

6502 DEBUG! FAST 'n EASY The PTD Language Way



```
05 LOC1 = $7C80
10 PC = $3FC7
20 LABEL: STEP 100 NODISP
30 IF X<$3E OR @LOC1#$17 THEN GOTO LABEL
40 PRINT "HERE IS THE CULPRIT"
50 SHOWC100
```

PTD-6502 is a high speed, compiled BASIC-like language, light years ahead of the Apple II Single Stepper and far more sophisticated than any other 6502 debugger available. It allows you to sit back effortlessly while your computer glides through your code at a thousand instructions per second looking for your bugs. Or you can select a slower speed with updated display of memory. A paddle-controlled single stepper mode is also available. At either of the slower speeds, the PTD-6502 monitors and saves the last 128 instructions executed for review at any time.

Virtually unlimited breakpoint complexity is permitted with the PTD-6502. IF statements with mixed AND's and OR's can be created to test conditions such as memory change, memory = value, instruction location, ... and many others. You can have as many named breakpoints as you wish in both ROM and RAM.

Some other features of the PTD-6502 include • Fast subroutine execution. • Hex calculator/converter. • Hex/ASCII memory dump. • Up to 16 machine language cycle timers. • Ability to monitor specific labeled areas in memory while stepping. • Effective address. • Accessible monitor commands. • A documented module for relocation of the PTD-6502 to virtually any location (source code supplied).

The debugging program shown on the monitor is a simple example; it could be far more complex. If you can think of it, you can probably scan for it at 1000 instructions per second. If you're a professional, the PTD-6205 can pay for itself in the first few hours of use. If you're a novice, you'll soon be debugging like a pro.

ORDER: PTD-6502 Debugger
including DOS 3.3 Disk
and instruction manual

\$49.95

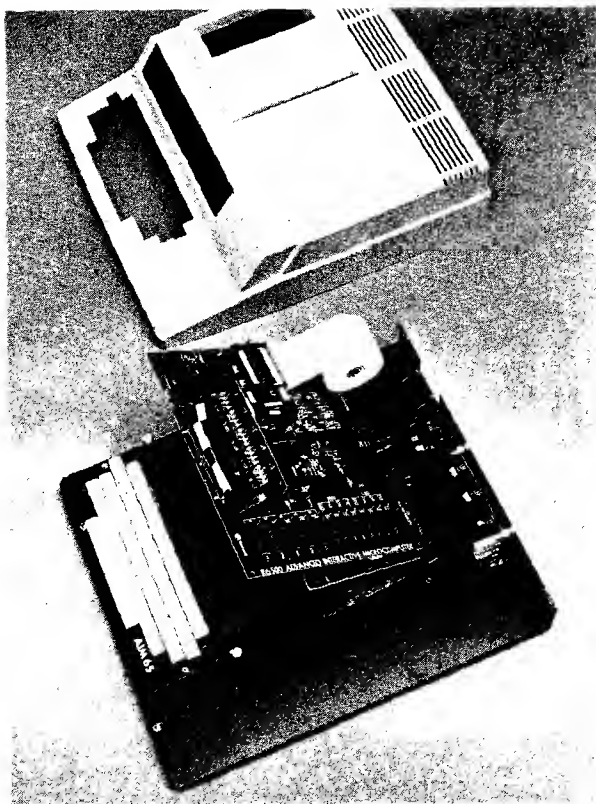
(Note that disk is not copy protected. Order only one for each business or institution.) In California, add 6.5% sales tax.

PTD-6502 requires Autostart ROM for fast breakpoint



**PTERODACTYL
SOFTWARE**™

1452 Portland Ave. • Albany CA 94706 • (415) 525-1605



AIM HIGH

Let Unique Data Systems help you raise your sights on AIM 65 applications with our versatile family of AIM support products.

- Go for high quality with our ACE-100 Enclosure. It accommodates the AIM 65 perfectly, without modification, and features easy access two board add-on space, plus a 3" x 5" x 17" and a 4" x 5" x 15.5" area for power supplies and other components. \$186.00.
- Get high capability with Unique Data System's add-on boards. The UDS-100 Series Memory-I/O boards add up to 16K bytes of RAM memory or up to 48K bytes ROM/PROM/EPROM to your Rockwell AIM 65. You also get 20 independently programmable parallel I/O lines with an additional user-dedicated 6522 VIA, two independent RS-232 channels with 16 switch-selectable baud rates (50 to 19.2K baud), and a large on-board prototyping area. Prices start at \$259.00.
- If you need to protect against RAM data loss, the UDS-100B offers an on-board battery and charger/switchover circuit. \$296.00.
- Heighten your AIM 65's communications range by adding the UDS-200 Modem board. It features full compatibility with Bell System 103 type modems and can be plugged directly into a home telephone jack via a permissive mode DAA. No need for a data jack or acoustic coupler. The UDS-200 also has software-selectable Autoanswer and Autodial capability with dial tone detector. The modem interfaces via the AIM 65 expansion bus, with the on-board UART and baud rate generator eliminating the need for an RS-232 channel. \$278.00.
- The UDS-300 Wire Wrap board accepts all 300/600/900 IC sockets from 8 to 64 pins. Its features include an intermeshed power distribution system and dual 44-pin card edge connectors for bus and I/O signal connections. \$45.00.
- Get high performance with the ACE-100-07 compact 4" x 5" x 1.7" switching power supply, delivering +5V @ 6A, +12V @ 1A, and +24V for the AIM printer. \$118.00.

Installation kits and other related accessories are also available to implement your AIM expansion plans. Custom hardware design, programming, and assembled systems are also available. High quality, high capability, high performance, with high reliability... all from Unique Data Systems. Call or write for additional information.

Unique Data Systems Inc.
1600 Miraloma Avenue, Placentia, CA 92670
(714) 630-1430

Reviews in Brief *(continued)*

registers. Disk tests include sequential and random writing and reading, random track seeking, and drive speed.

Pluses: Single or multiple tests may be repeated continuously, with results optionally printed. The program is menu-driven, user-friendly, fast, and crash-resistant.

Minuses: The style and depth of the documentation are marginal.

Documentation: The 24-page manual is neatly formatted and printed. The writing is comprehensible but often awkward and unpolished. It describes in detail how to use the program, but gives almost no help to analyze and correct problems it finds.

Skill level required: Little skill is needed to run it, but moderate hardware knowledge is required to know what to do about reported problems.

Reviewer: Jon R. Voskuil

Product Name: SPELL 'N FIX
Equip. req'd: TRS-80C, with disk or cassette, 32K, other versions available for FLEX, OS-9, and other systems.
Price: \$69.29 (FLEX version \$89.29)
Manufacturer: Star Kits
P.O. Box 209
Mt. Kisco, NY 10549

Description: SPELL 'N FIX is a package of program files that provides a dictionary for Color Computer text files. The main program, SPELLFIX, loads and executes a 6809 machine-language dictionary look-up program. A 20,000-word dictionary file is used to check ASCII files for spelling and typographical errors. Other files included are utilities for writing and reading ASCII files, a sample text file, binary-to-ASCII conversion programs, and a program to expand the dictionary. These programs allow you to use SPELLFIX with processors that create binary files.

Pluses: The dictionary program is expandable when using the disk version, and you can create your own dictionary that fits your writing style. Questionable words are displayed, and/or printed in alphabetical order for checking. The disk version also allows marking of questionable words for later correction, or they may be corrected immediately. Large files usually take only slightly longer to correct than smaller files. It will work on most files that are larger than RAM memory. The disk version can be easily converted to tape, and vice versa.

Minuses: The tape version cannot mark or immediately correct text files. Not directly compatible with Color Scripsit files, though, Scripsit can print an ASCII file to tape, which can be read by the dictionary.

Documentation: A 25-page manual is included that thoroughly explains the proper operation of the programs. Information is also provided on modifying and creating new dictionaries. No instructions were included for removing words from the dictionary.

Skill level required: With only a few minutes of study, anyone should be able to operate the program.

Reviewer: John Steiner

MICRO

Software Catalog

Name: **Data Tape Maker**
System: OSI
C1P/Superboard II
Memory: 4K
Language: 8K BASIC in ROM
Description: *Data Tape Maker* is a relatively short program that allows you to save machine-language code or any other data stored in consecutive memory locations in DATA statements on tape. The sign space for each number is eliminated to allow for compact storage of data. A FOR/NEXT loop is automatically generated to restore the data into memory at a later time.
Price: \$4.00 for tape
\$3.00 for listing
Author: Brian Zupke
Available:
B.C. Software
5152 Marcell Ave.
Cypress, CA 90630

Name: **Air Navigation Trainer**
System: Apple II or Apple II Plus, Applesoft in ROM or Language Card
Memory: 48K
Language: Applesoft and Machine Language
Hardware: One disk drive (DOS 3.3) and game paddles
Description: *Air Navigation Trainer* is a real-time simulation of aircraft navigation with hi-res instrumentation and ground-track map, sound effects (including station IDs), dial-in wind magnitude and direction, four different simulations, dual independent VORs (VHF Omnirange Radar) with adjustable OBS (just like the real thing), ADF, NDBs, and more.
Price: \$40.00
Includes program diskette and full documentation.
[Not for pilots only!]
Author: Ken Winograd
Available:
Space-Time Associates
20-39 Country Club Drive
Manchester, NH 03102
(603) 625-1094

Name: **Spellmaster (ProofReading Software)**
System: CBM 8032, CBM 8096, SuperPET, Commodore 64
Memory: 32K minimum
Language: Assembly [6502]
Description: *Spellmaster* identifies and allows correction of misspellings from wordprocessing text. It has a 40,000-word capacity on the CBM 8050. Suspect words are displayed on screen, and direct screen editing of mistakes is provided. Available for WordPro, Wordcraft, Silicon Office. It will proofread a large WordPro file in two minutes or less. Legal and medical dictionaries are available for \$75.
Price: \$199.00
Author: Dwight Huff and Joe Spatafora
Available:
Spellmaster Systems
Software
6219 13th Avenue South
Gulfport, FL 33707
(813) 347-6733

Name: **Rail Runner**
System: TRS-80 Color Computer or TDP System 100
Memory: 16K
Language: Assembly
Hardware: Cassette or disk
Description: Your railroad engineer must scurry over the track of the busiest train switchyard ever, dodging speeding trains and handcars, to rescue the poor little hoboes on the wrong side of the tracks. You have only so much time to save all the hoboes! With many levels of difficulty, this action graphics game is fun for everyone.
Price: \$21.95 cassette
\$26.95 disk
plus \$2 shipping
Includes cassette or disk with instructions.
Author: BJ
Available:
Computerware
Box 668
Encinitas, CA 92024
(714) 436-3512

Name: **K-Star Patrol™**
System: Atari 400/800
Memory: 8K
Language: Machine Code
Hardware: ROM cartridge
Description: An exciting galactic encounter between the player's patrol flight and an onslaught of attacking alien craft. The player's mission is further complicated by a voracious intergalactic leech, and the aliens' low-level avoidance system. High degree of challenge and entertainment for even the most experienced player.
Price: \$39.95 suggested retail
Includes ROM cartridge and full color instruction booklet.
Author: Dr. Keith Dreyer and Torre Meeder
Available:
K-Byte
1705 Austin
Troy, MI 48084
or your local computer software retailer

Name: **Death Race '82**
System: Apple II with Applesoft in ROM
Memory: 48K
Language: BASIC/Assembler
Hardware: One disk drive, game paddles
Description: *Death Race '82* combines the skill of perilous driving with the thrill of a high-speed chase. Behind you is a robot car fully equipped with high-technology lasers. Your successful escape depends on maneuvering your turbo car through the enigmatic curves of ten consecutive mazes, and foiling your pursuer through the clever use of bazooka rockets and oil slicks. Ten different speeds ranging from novice to expert offer hours of fun before proficiency is achieved.
Price: \$29.95
Includes disk and documentation.
Author: Don Fudge
Available:
Avant-Garde Creations
P.O. Box 30160
Eugene, OR 97403
or local dealers

Name: **Single Entry Ledger**
System: 6809 Using FLEX or UniFLEX, TRS-80 Model III and Color Computer
Memory: 56K
Language: Extended BASIC
Hardware: 8" or 5¼" disk
Description: *Single Entry Ledger* is a simple bookkeeping system for tracking income and expenses. It is an ideal accounting system for tax purposes saving the user both time and money. The data files may contain any number of accounts or transactions. Any number of reports may also be written from comparison reports of the previous year to transactions by account number.
Price: \$95.00
Includes disk and manual.
Author: K. Orlowski
Available:
Universal Data Research Inc.
Dept. A
2457 Wehrle Drive
Buffalo, NY 14221

Name: **Prelab Studies in General Organic and Biological Chemistry**
System: Apple II with 3.3 DOS
Memory: 48K
Language: Applesoft
Description: This package provides a review of selected chemical concepts highlighting important ideas, techniques, and calculations encountered in the laboratory. The programs are in a tutorial format, using demonstrations, interactive exercises, animated sequences, and simulations.
Price: \$550.00 (tentative)
Includes nine disks and complete documentation.
Author: Sandra L. Olmsted and Richard D. Olmsted
Available:
John Wiley & Sons, Inc.
Eastern Distribution Center
Order Processing
Department
1 Wiley Drive
Somerset, NJ 08873

Software Catalog *(continued)*

Name: **System/ASM 3A**
System: Apple II Plus
Memory: 48K minimum.
Language card is supported.
Language: Assembly
Hardware: Disk II required, Silentype printer optional

Description: *System/ASM 3A* is an assembly-language development system that features a two-pass assembler, full screen editor, and disk-file management system. The system is easy to use but powerful enough to write very complex programs. *System/ASM 3A* is written in its own assembly language and is DOS 3.3-compatible.

Price: \$35.00
\$5.00 for manual only
Includes no shipping and handling charges. Ohio residents add appropriate sales tax.

Available:
The Mike Piaser Company
15401 Maple Park Drive #11
Maple Heights, OH 44137

Name: **Factoring Whole Numbers**
System: PET DOS 2.1
Memory: 16K
Language: BASIC
Hardware: Disk drive or cassette

Description: Twelve programs (on six tapes or three diskettes) present the concepts of factoring in a carefully-designed sequential preparation for fractions and algebraic expressions. A tutorial and practice program precedes six motivating and interactive enrichment programs.

Price: \$90.00
Includes diskettes or tapes and a teacher's guide.

Author: Joanne Benton
Available:
Quality Educational Designs
P.O. Box 12486
Portland, OR 97212

Name: **Android Attack**
System: Atari 400/800
Memory: 16K cassette
32K disk
Language: Hybrid
Hardware: Cassette or disk system
Description: The nuclear reactor in our top-secret underground lab is in danger of melting down! Only you can save it by manually releasing

the coolant water. Unfortunately, there isn't time to disarm the security Androids guarding the installation, so you'll have to fight your way down. Once you've released the water, you've got to get back out before you drown! *Android Attack* has electric robots and walls, bonus points, and up to eight different levels to challenge you!

Price: \$18.95 plus \$2 shipping (Mail order price)

Author: John Wilson

Available:
Pretzelland Software
2005 D. Whittaker Rd.
Ypsilanti, MI 48197
(313) 483-7358
or local dealers

Name: **The Last One**
System: Apple II Plus
Memory: 48K
Language: BASIC/Machine
Hardware: Two disk drives, printer optional

Description: *The Last One* is a computer program code generator that designs a program and enters flowchart-type statements in an easy-to-use menu style. *The Last One* then begins to code the program, asking the user questions about "where to branch," etc. A BASIC program is created as output which then can be run, listed, or modified like any other BASIC program. *The Last One* is not required to execute the output program.

Price: \$600.00
Includes complete documentation, numerous sample flowcharts that will produce software worth several hundred dollars.

Author: D.J. 'Al' Systems Ltd.
Available:
Krown Computing
1282 Conference Dr.
Scotts Valley, CA 95066
(408) 335-3133

Name: **Assemblers Package I**
System: The UCSD p-System™
Memory: 48Kb runtime environment; 64Kb development environment
Language: Assembly
Hardware: 8086, Z80, 8080, 8085, 6502, 9900, 6809, 68000, and LSI-11/PDP-11

Description: This collection of native code-generating macro cross-assemblers allows you to program on the host machine of your choice for the object machine of your choice.

Price: \$375.00
Includes object code.

Available:
SofTech Microsystems, Inc.
9494 Black Mountain Rd.
San Diego, CA 92126
(714) 578-6105

Name: **Galactic Gladiators**
System: Apple II with Applesoft ROM card, Apple II Plus, or Apple III
Memory: 48K
Hardware: Monitor and disk drive

Description: *Galactic Gladiators* is a fast and furious computer game of alien combat for two players or against the computer. The creatures are rated for strength, endurance, speed, dexterity, experience, weapons, skill, and armor. The scenario permutations are as infinite as the Universe.

Price: \$39.95
Includes rulebook, disk, and data card.

Author: Tom Reamy
Available:
Strategic Simulations Inc.
465 Fairchild Dr.
Suite 108
Mountain View, CA 94043
(415) 964-1353

Name: **The Animator**
System: Apple II or Apple II Plus
Memory: 48K
Language: Applesoft/Assembly
Hardware: Disk drive
Description: This program produces animated 'film' strips that enter only key frames, then *The Animator* calculates the in-between frames. The key frames are easily entered — either visually, numerically, or from a library. The demo includes a ballet sequence showing a ballerina with 12 independently moving body parts.

Price: \$51.95
Includes 57-page manual, three tutorials, and a shape generator.

Author: Ray Balbes
Available:
Balbesoftware Systems
#6 White Plains Dr.
St. Louis, MO 63017
(314) 532-5377

Name: **The Apple Family Sing-Along Christmas Disk**
System: Apple II, Apple II Plus, Apple III
Memory: 48K
Language: Applesoft or Integer Basic (runs in emulation mode on Apple III)

Hardware: Disk drive
Description: Sixteen favorite carols, complete with words to all the verses, containing multiple-voices and four-part harmony, are pitched so you can sing along if you want to. The choice of an internal speaker or cassette port output is given. The Christmas music is tuneful, well arranged, and lots of fun to listen to. Just the thing to lend novelty and a festive background to Christmas parties, office parties, and Apple family get-togethers.

Price: \$24.50
Includes everything needed to play the songs — no hardware required.

Author: Product of the *Music Maker™* utility from SubLogic Communications Corp.

Available:
Solutions Softworks
Box 72280
Roselle, IL 60172
\$1.50 shipping costs or from Apple dealers

Name: **Anova II**
System: Apple II or Apple II Plus
Memory: 48K
Language: ROM Applesoft
Hardware: One or two disk drives, printer optional

Description: *Anova II* performs up to a five-way analysis of variance with equal or unequal numbers. It can analyze randomized designs, between and within designs, and repeated measures of designs. *Anova II* can also perform an analysis of co-variance for all designs. The *Anova* table output tests all factors and interactions.

Price: \$150.00
Includes program disk and backup disk, documentation, and binder.

Authors: Stephen Madigan, Ph.D. and Virginia Lawrence, Ph.D.

Available:
Human Systems Dynamics
9249 Reseda Blvd.
Suite 107
Northridge, CA 91324

(continued)

Name: **UniFLEX**
 System: Gimix 6809 Winchester Systems
 Memory: 128K minimum
 Language: Available: BASIC, Pascal, Assembler, FORTRAN 77, C
 Hardware: 2MHZ 6809 CPU with memory, disk controllers, 19MB 5¼" Winchester

Description: *UniFLEX* is a true multi-tasking, multi-user operating system. Each user communicates with the system through a terminal and may execute any of the available system programs. This implies that one user may be running the text editor while another is running BASIC while still another is running the C compiler. Not only may different users run different programs simultaneously, but one user may be running several programs at a time.

Price: \$550.00
 Includes UniFLEX Operating System, documentation.

Author: Technical Systems Consultants, Inc.

Available:
 Gimix Inc.
 1337 W. 37th St.
 Chicago, IL 60609
 (312) 927-5510

Price: \$99.95/Sinclair tape
 \$129.95/Apple/Atari disk
 \$129.95/Atari tape
 Includes 34 pages of documentation.

Author: Bob Nadler

Available:
 F/22 Press
 P.O. Box 141
 Leonia, NJ 07605

Name: **Lovers or Strangers**
 System: Apple II
 Memory: 48K
 Language: Applesoft
 Hardware: One disk drive
 Description: *Lovers or Strangers* is a computer game with a serious side. It is a compatibility evaluator that tells two people how likely they are to have a successful relationship. A couple's likes and dislikes, philosophies, and lifestyles in seven major areas of compatibility are explored.

Price: \$29.95
 Includes program disk and written instructions.
 Author: Stanley Crane
 Available:
 Alpine Software, Inc.
 2120 Academy Circle, Suite E
 Colorado Springs, CO 80909
 (303) 591-9874

Name: **The Football Comput-Stat**
 System: Apple II, IBM PC, Radio Shack MIII
 Memory: 48K
 Language: BASIC
 Hardware: One disk drive, printer optional

Description: *Compu-Stat* contains programs and related data for the analysis of pro-football's regular season — both point-spread records and the underlying box-score statistics. It performs analyses for the 1981 and 1982 regular seasons. A related program product, Tally Sheet, keeps a running tally on your predictions.

Price: \$100 - \$3500 depending on programs and equipment ordered.

Includes user manual, program diskette, and security chip.

Author: Dr. John Page

Available:
 Interactive Sports Systems
 P.O. Box 15952
 New Orleans, LA 70175

Name: **Elements of Mathematics**
 System: Apple II
 Memory: 48K
 Language: BASIC
 Hardware: One disk drive
 Description: This program was developed to assist students in adding fractions, reducing fractions, and adding fractions with unlike denominators. Materials were developed and tested by the authors before being published.

Price: \$90.00

Author: Ray E. Zubler
 Susan Sarapata

Available:
 Electronic Courseware Systems, Inc.
 P.O. Box 2374, Station A
 Champaign, IL 61820
 (217) 359-7099
 or computer retail stores and book stores

(continued)

What's eating your Apple?

Find out with Apple-Cillin II™

If you use your Apple for your business or profession, you probably rely on it to save you time and money. You can't afford to guess whether it is working properly or not. Now you don't have to guess. Now you can find out with Apple-Cillin II.

Apple-Cillin II is the comprehensive diagnostic system developed by XPS to check the performance of your Apple II computer system. Apple-Cillin II contains 21 menu driven utilities including tests for RAM memory, ROM memory, Language Cards, Memory Cards, DISK system, Drive Speed, Keyboard, Printer, CPU, Peripherals, Tape Ports, Monitors and more. These tests will thoroughly test the operation of your Apple, and either identify a specific problem area or give your system a clean bill of health. You can even log the test results to your printer for a permanent record.

Apple-Cillin II works with any 48K Apple system equipped with one or more disk drives.

To order Apple-Cillin II - and to receive information about our other products - Call XPS Toll-Free: 1-800-233-7512. In Pennsylvania: 1-717-243-5373.

Apple-Cillin II: \$49.95. PA residents add 6% State Sales Tax.



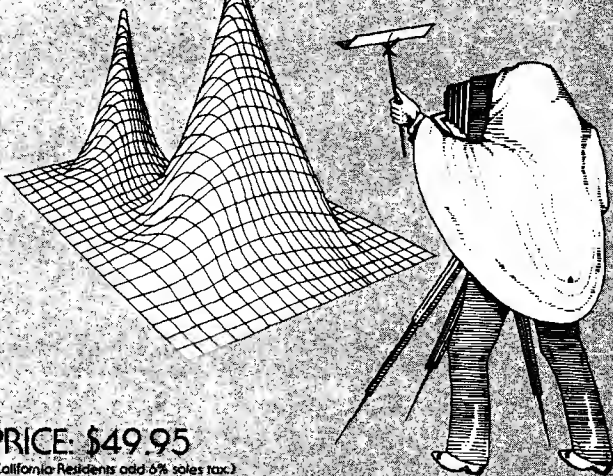
XPS, Inc.
 323 York Road
 Carlisle, Pennsylvania 17013
 800-233-7512
 717-243-5373

Apple II is a trademark of Apple Computer Inc.

**TIRED OF ALL THE
"EXCEPT FOR..."S?**

**WITH THE NEW PRINTOGRAPHER
GRAPHICS PRINTING SYSTEM,
WE'VE GOT YOU COVERED!**

PRINTOGRAPHER



PRICE: \$49.95

(California Residents add 6% sales tax.)

The PRINTOGRAPHER is designed to fill all of your graphic printing needs, without having to worry about running into the problem of it almost working, "except on your printer", or "except for the lack of that particular feature". Whether you have a daisy wheel or dot matrix printer, the standard version of PRINTOGRAPHER works on any printer and interface combination with graphics capabilities. In many cases, this includes printers you may not even have thought could print graphics.

Just a FEW of the possible printers include: EPSON, PAPER TIGER, ANADEX, NEC, DIABLO, QUME, MPI, SILENTYPE, OKIDATA, MALIBU; interface cards include: APPLE, SSM, CCS, MTN COMP, CPS, MPI, GRAPPLER, TYMAC, PROMETHEUS and more!

In addition to versatile print options (easy cropping, variable magnifications, normal/reverse inking, vertical/horizontal format, etc.) PRINTOGRAPHER offers such unique features as the ability to print pictures directly from disk (without loading a file), spooling via our DOUBLETIME PRINTER package, or sending pictures over a phone line using ASCII EXPRESS. You can even put graphics in your text documents with our text editor software, THE CORRESPONDENT. As if that wasn't enough, we've made it easy to put the PRINTOGRAPHER routines right in your own programs to do Hi-Res printing immediately during their operation, without having to save screen images to disk!

We also know you see a lot of advertising these days for a truly overwhelming volume of software, all claiming to be the best, so we make this simple guarantee:

IF YOU CAN FIND A BETTER PACKAGE THAN (OR ARE AT ALL UNSATISFIED WITH) THE PRINTOGRAPHER WITHIN 30 DAYS OF PURCHASE, SIMPLY RETURN THE PACKAGE FOR A COMPLETE REFUND. NO QUESTIONS ASKED!

For more information, see your local dealer, or write SOUTHWESTERN DATA SYSTEMS for a free catalog. If your dealer is out of stock, we can ship PRINTOGRAPHER to him within 24 hours of a call to our offices.

REMEMBER: WITH PRINTOGRAPHER, YOU'RE PICTURE PERFECT!

southwestern data systems

10761 E Woodside Avenue • San Jose, California 95071
Telephone: 714/562-3670

Software Catalog (continued)

Name: Basic Aid
System: TRS-80 Color Computer
Memory: 16K - 64K
Language: 6809 Machine Language
Hardware: ROMPAK
Description: Basic Aid is a utility program to help and assist Color BASIC and Extended BASIC users. Some of the features are: automatic line numbering, program merging, and moving program segments. It comes with a plastic keyboard overlay that contains most of Extended Color BASIC's commands.

Price: \$34.95
Includes detailed instruction manual, plastic keyboard overlay.

Author: Eigen Systems

Available:
Spectrum Projects
93-1586 Drive
Woodhaven, NY 11421

Name: S-C Macro Cross Assemblers 6800, 6809, and Z-80

System: Apple II or Apple II Plus

Memory: 48K (RAM card version included)

Language: Machine

Hardware: Disk drive

Description: You can easily develop programs for 6800, 6809, or Z-80 computers with powerful macros, conditional assembly, 20 directives, and 29 commands (including a powerful EDIT command with 15 subcommands). It allows very fast cycles of modification, reassembly, and testing.

Price: \$110.00 each.
Registered owners of the S-C Macro Assembler pay \$32.50 each.

Includes diskette with regular and RAM card versions, 100+ page manual.

Available:
S-C Software Corporation
P.O. Box 280300
2331 Gus Thomasson
Suite 125
Dallas, TX 75228
(214) 324-2050

Name: GL-PLUS
System: Apple III
Memory: 128K
Language: Business BASIC
Hardware: 132-column printer and either second diskette drive or hard drive.

Description: GL-PLUS is an extremely flexible and easy to

operate general ledger with built in receivables and payables. Reports include general ledger, month's journal, balance sheet, income statement, aged receivables and payables, receivable and payable detail, and more!

Price: \$495.00

Includes operator's manual, programs, and sample company data.

Author: Dan Sargent

Available:
Great Divide Software
8060 W. Woodard Dr.
Lakewood, CO 80227

Name: Borg
System: Apple II or Apple II Plus

Memory: 48K

Language: Assembly

Hardware: One disk drive, paddle or joystick

Description: Deranged Grud Terrorizes Countryside! Protected by Borg, the invincible Drageroo, a notorious band of dragons, the infamous Grud has surrounded his hide-out with electrified mazes. Can no one crack the code and rid us of this menace?

Price: \$29.95

Author: Dan Thompson

Available:
Sirius Software, Inc.
10364 Rockingham Dr.
Sacramento, CA 95827
(916) 366-1195

Name: D.F.T
System: TRS-80 Color Computer

Memory: 16K

Language: Machine

Hardware: Cassette recorder
Description: This terminal program allows you to download any type of program — BASIC or machine language — or ASCII with no conversion. It allows transfer of programs between TRS-80 Mod I's, Mod III's, and the Color Computer.

Price: \$19.95

Includes one tape.

Author: Bob Withers

Available:
Computer Shack
1691 Eason
Pontiac, MI 48054

Correction: The software listing for Jinsam Executive (52:116) from JINI Micro-Systems, Inc., should have read 32K for CBM w/8050, and 128K IBM PC for BASIC and machine language. It is available from the company and participating dealers.

MACRO

Hardware Catalog

Name: **Guild Computer Rack**

System: Apple II
Description: The *Guild Rack* comes in a choice of beautifully finished mahogany or ash. No assembly is required. It fits comfortably over the Apple II keyboard, holds one or two disk drives, and easily supports a monitor on top.

Price: \$54.95 - ash
\$69.95 - mahogany

Available:

Guild Computer Rack
225 West Grand Street
Elizabeth, NJ 07202
(201) 351-3002

Name: **Disk Interface/
ROMpak
Extender**

System: Color Computer
Memory: 4K and up
Hardware: Three-foot
extender cable

Description: The *Disk Interface/ROMpak Extender* is a 40-pin ribbon cable that plugs into the ROMpak port and terminates three feet later with a 40-pin female connector to connect ROMpaks and the disk interface. Gold-plated contacts eliminate corrosion.

Price: \$29.95 plus \$1 for S/H
Includes male and female connector, three feet of 40-conductor cable.

Available:

Spectrum Projects
93 - 1586 Drive
Woodhaven, NY 11421
(212) 441-2807 Voice
(212) 441-3755 Computer

Name: **Versaclock**

System: TRS-80 Color
Computer
Memory: 4K and up
Language: BASIC or
Extended BASIC

Description: The *Versaclock* is a full-featured, highly accurate hardware clock for the Color Computer. It provides time of day, date, month, and year with automatic daylight savings time and leap year compensation. The clock is battery backed-up to allow removal from computer without loss of data. The clock also contains 50 bytes of battery backed-up RAM for general purpose per-

manent storage. The many software options include interrupt handling and 12/24 hour formats.

Price: \$99.95

Includes Versaclock cartridge, full instructions.

Available:

Maple Leaf Systems
Box 2190
Station "C", Downsview
Ontario, Canada M2N-2S9

Name: **Color Graphic
Printer (26-1192)**

System: Compatible with
TRS-80 Models I,
II, III, and Model
16 computers, and
DT-I Data
Terminal

Description: The TRS-80 *Color Graphic Printer* can create anything from doodles to four-color pie charts, as well as more standard text and graphics. Ninety-six ASCII characters are available in four colors (red, blue, green, black). Special graphic commands include backspace, reverse line feed, change colors, change line type (solid or 15 types of dashed lines), change print direction [normal left-to-right, top-to-bottom, upside down or bottom-to-top], move without drawing, draw between points and draw axes. The RS232-C serial interface is compatible with Radio Shack TRS-80 Color Computers.

Price: \$249.95

Available:

Radio Shack Stores,
computer centers, and
participating dealers

Name: **K-Byte Stick
Stand with
Fastball Easy-Grip
Control Knob.**

Description: K-Bytes unique *Stick Stand with the Fastball Easy-Grip Control Knob* reduces hand and wrist fatigue and frees one hand for a more skillful operation of the firebutton. This combination allows players to increase their physical dexterity and achieve higher scores. By just snapping the fastball onto the joystick and then snapping the joystick into the stick stand, the player

is all set for precision arcade action.

Price: \$6.99 suggested retail
Includes base stand and
fastball knob.

Available:

John Mathias
K-ByteTM
Div. of Kay Enterprises Co.
P.O. Box 456
1705 Austin
Troy, MI 48099
(313) 524-9878
or your local computer
retailer

Name: **Fast Load — Fast
Save Cassette
System**

System: OSI - C1P or
Superboard II

Description: Load BASIC or machine-language programs in your 8K memory in less than 30 seconds at a speed of 2400 bits per second input/output data rate. Customer supplies own tape recorder. The unit includes a 2K RAM fully decoded which may be used to hold machine-language programs. Unit plugs directly into your C1P or Superboard II.

Price: \$69.95 fully assembled
\$59.95 with cashier's check
or money order.

\$62.95 kit
\$52.95 with cashier's check
or money order.

Includes printed circuit
board, cassette tape program,
self-contained R/W memory,
connectors, and user's
manual.

Available:

Word-Com
P.O. Box 1122 - 28
Park Plaza Offices
303 Williams Ave.
Huntsville, AL 35801

Name: **Pro-Guard 8"
Floppy Controller**

System: Apple III
Memory: Up to 2.2
megabytes

Language: SOS, DOS 3.3,
Pascal

Hardware: Controls two 8"
Shugart-
compatible drives

Description: This 8" floppy controller resides in-line between Apple III and the drive system and connects to slot 2

via SVA's innovative *Smart-Cable*.

Price: \$695.00

Available:

SVA Sorrento Valley
Associates, Inc.
11722 Sorrento Valley Rd.
San Diego, CA 92121
Apple dealers, Micro-D,
Micro House, U.S. Micro
Sales

Name: **Ramex 128**

System: Apple II or Apple
II Plus

Memory: 48K

Description: This 128K RAM expansion board includes disk-emulation software that features super-fast mounts and dumps from card to disk (20-25 seconds for an entire 128K). Also available for VisiCalc is super expander software that gives the same super-fast loading and saves of VisiCalc files (136K in 20 seconds).

Price: \$499.00

Includes disk emulation
software and memory
management.

Available:

Omega Microware, Inc.
222 S. Riverside Plaza
Chicago, IL 60606

Name: **Multi-Port 232**

Description: The *Multi-Port 232* is a 4- or 8-port multidrop data router that allows merging or splitting of RS232, fiber optic, and current loop in any source/destination combination. It provides local networking for word processors, printers, modems, video displays, computers, teletypes, and instruments.

Price: \$435.00 - 4-port

VISA/Master Charge
Includes nine user-selectable
preprogrammed routes.

Available:

Park Computer Corporation
Box 13010
Minneapolis, MN 55414

6809 Bibliography

86. Color Computer News, Issue No. 11 (August, 1982)

- Ostrom, Steven M., "Graphics and Animation for the Color Computer," pg. 30-42.
A tutorial for the TRS-80 Color Computer graphics with a number of demo routines.
- Dawson, Don, "Color Yahtzee," pg. 44-47.
A game for the 6809-based Color Computer.
- Phelps, Andrew, "Comment Corner," pg. 49-50.
A tutorial on RAM hooks, places where the program jumps, and which then jump elsewhere in memory.
- McClenahan, Shawn A., "A Real Keyboard for the Color Computer," pg. 55-60.
A hardware project for the Color Computer.
- Field, E.C., "Electro-Sketch," pg. 67-69.
A graphics program for the 6809-based Color Computer which allows one to draw simple schematics and save or print them.
- Lee, Paul, "Educating Your Preschooler with the Color Computer," pg. 71.
A simple teaching program for young children using the Color Computer.
- Weiss, Arnold, "Cryptogram," pg. 72-76.
A program to present cryptograms on the TRS-80 Color Computer screen or to make printed copies.
- Harper, Jeff, "Word Processor," pg. 77-79.
A word-processor program for the 16K or 32K Extended BASIC Color Computer.
- Foster, Robert D., "Monitor," pg. 81-82.
A simple monitor to allow one to see how the Color Computer actually works.
- Tenny, Ralph, "Extra Tricks with Color Scripsit," pg. 84-85.
An accessory listing to aid in using Color Scripsit.
- Aldrich, F.C., "Magic Square," pg. 87-89.
A contest-winning listing for the 6809-based Color Computer.

87. '68' Micro Journal, 4, Issue 8 (August, 1982)

- Anderson, Ronald W., "FLEX User Notes," pg. 11-14.
Miscellaneous notes on FLEX for the 6809-based systems. Includes a multiply program in assembly language.
- Nay, Robert L., "COLOR User Notes," pg. 14-16.
Discussion of some new items for the 6809-based Color Computer.
- Abrams, Clayton W., "F-Mate," pg. 16-17.
A utilities package for the TRS-80 Color Computer.
- Distefano, Tony, "Color Clinic," pg. 17-18.
Discussion of hardware modifications for the TRS-80 Color Computer.
- Commo, Norm, "'C' User Notes," pg. 19-24.
Discussion of major C compilers for 6809 systems.
- Watson, Ernest Steve, "Home Accounting Program — Part II," pg. 25-28.
A program for 6809 systems.
- Hartman, William, "Diskfix9," pg. 29-36.
A utility for 6809-based systems.

88. The Rainbow, 2, No. 2 (August, 1982)

- Nolan, Bill, "Let's Call JOYIN To Learn ROM Call Technique," pg. 8.
A short program illustrating how to call one of the built-in ROM routines in the TRS-80 Color Computer.
- Lishnak, Pat, "Sort Numeric Arrays Fast with Machine Language," pg. 9.
A bubble sort technique for the Color Computer.

Boston, William, "Here's An Easy Way to Place Orders by Mail," pg. 19-20.

An order-writing program for the Color Computer.

Lewandowski, Dennis S., "The Assembly Corner," pg. 22-25.

A tutorial on 6809 assembly-language programming.

Clements, Bill, "Rockin' Through the ROM," pg. 29-30.

Documenting the ROM routines of the TRS-80 Color Computer.

Preble, Laurence D., "FLEX System is Powerful Addition to World of 80C," pg. 32-33.

All about the FLEX system for the 6809 micro.

Scerbo, Fred B., "Alpine Aliens," pg. 34-37.

A game for the Color Computer.

Blyn, Steve, "Good Reinforcement Means You Can't Frown at Me!," pg. 41-46.

Tips and demo program for educational use.

Mir, Jorge, "Now, Make Your Own Adventure with ADVMAKER," pg. 47-53.

A program designed to simplify the programming of Adventures written for the Color Computer.

Nolan, Bill, "Dragons Are Nice Folks, Too... Almost All 1,440 of Them," pg. 62-69.

The program "Dragon Roller" will assist with the chore of devising a dragon for your dungeon program.

89. Byte, 7, No. 8 (August, 1982)

Williams, Gregg, "LOGO for the Apple II, the TI-99/4A, and the TRS-80 Color Computer," pg. 230-290.

Discussion of LOGO for several micros, including the 6809-based Color Computer.

90. The Target (March/April, 1982)

Staff, "News," pg. 1.

An assembly which converts an AIM 65 into a 6809-based computer.

91. Compute! 4, No. 8 (August, 1982)

Chastain, Linton S., "Energy Monitor," pg. 116-118

This program for the TRS-80 Color Computer will show you the effects of home energy conservation.

92. Microcomputing, 6, No. 9 (September, 1982)

Avery, Mike, "Prime Number Nonsense," pg. 16.

Comments on the 6809 versus the 6502, Z-80, or 6800 microprocessors.

93. Color Computer News (August, 1982)

Sias, Bill, "REMARKS," pg. 6-7.

Announcement of the 6809 Achievement Award being given monthly to the most innovative use for a 6809.

Gray, Don, "Number Conversion," pg. 7-8.

Three listings for number conversion programs for the 6809-based TRS-80 Color Computer.

Anon, "Color Computer Bulletin Board System," pg. 11.

A BBS for the Color Computer is up in the Toronto area. Call (416) 494-7001 evenings and weekends.

Donahue, Mike, "Cross-Reference Generator," pg. 15-25.

A utility for the 6809-based TRS-80 Color Computer.

Grady, Larry, "Review of Master Control," pg. 29-33.

Problems encountered with the program "Master Control" and some reprogrammed sections to alleviate difficulties.

Graham, Randy W., "Modems, Terminals, and Bulletin Boards," pg. 35-38.

Using the Color Computer in telecommunications.

COMMODORE 64

The Commodore 64 is a 6510-based color-and-sound computer that connects to a color TV via an RF modulator. 64K RAM is standard, with 39K of it available for BASIC programs.

Graphics

- 3 character modes
- 2 bit-map modes
- sprite graphics

Sound

- 4 programmable voices
- attack, sustain, decay, and release
- output compatible with stereos

Z-80 option for CP/M

RS-232, expansion/cartridge, parallel, cassette and controller interfaces

Commodore 64 Memory Map

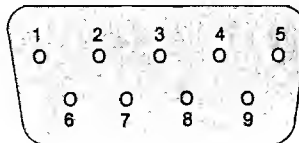
Address	Function
\$00-\$FF	Page zero, operating system storage, pointers, floating point accumulators, flags, etc.
\$100-\$1FF	Microprocessor system stack
\$100-\$10A	Floating-to-string work area
\$100-\$13E	Tape input error log
\$200-\$2FF	Operating system buffers, tables, vectors, I/O flags, keyboard handling
\$300-\$3FF	Vectors, tape I/O
\$400-\$7FF	Normally video memory, sprite data pointers, etc.
\$800-\$9FFF	Normally BASIC program space
\$8000-\$9FFF	VSP Cartridge ROM
\$A000-\$BFFF	BASIC ROM
\$C000-\$CFFF	RAM
\$D000-\$DFFF	I/O devices and color RAM or character-generator ROM
\$E000-\$FFFF	Kernal ROM

Control Port 1

Pin	Function
1	JOYA0
2	JOYA1
3	JOYA2
4	JOYA3
5	POT AY
6	BUTTON A/LP
7	+5V
8	GND
9	POT AX

Control Port 2

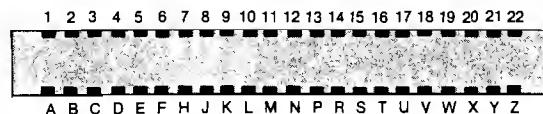
Pin	Function
1	JOYB0
2	JOYB1
3	JOYB2
4	JOYB3
5	POT BY
6	BUTTON B
7	+5V
8	GND
9	POT BX



Cartridge Expansion Slot

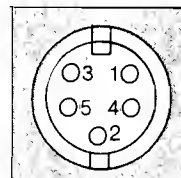
Pin	Function
1	GND
2	+5V
3	+5V
4	IRQ
5	CR/W
6	Dot Clock
7	I/O1
8	GAME
9	EXROM
10	+I/O2
11	ROML
12	BA
13	DMA
14	D7
15	D6
16	D5
17	D4
18	D3
19	D2
20	D1
21	D0
22	GND

Pin	Function
A	GND
B	ROMH
C	RESET
D	NMI
E	S02
F	A15
H	A14
J	A13
K	A12
L	A11
M	A10
N	A9
P	A8
R	A7
S	A6
T	A5
U	A4
V	A3
W	A2
X	A1
Y	A0
Z	GND



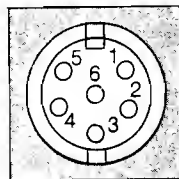
Audio/Video

Pin	Function
1	LUMINANCE
2	GND
3	AUDIO OUT
4	VIDEO OUT
5	AUDIO IN



Serial I/O

Pin	Function
1	SERIAL SRQIN
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	RESET



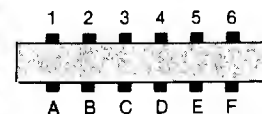
User I/O

Pin	Function
1	GND
2	+5V
3	RESET
4	CNT1
5	SP1
6	CNT2
7	SP2
8	PC2
9	SER. ATN IN
10	9 VAC
11	9 VAC
12	GND

Pin	Function
A	GND
B	FLAG2
C	PB0
D	PB1
E	PB2
F	PB3
H	PB4
J	PB5
K	PB6
L	PB7
M	PA2
N	GND

Cassette

Pin	Function
A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SENSE



COMMODORE 64

MICRO™ Data Sheet #11

COMMODORE 64

MOS 6566 Video Interface Controller (VIC II)

Hex	Decimal	Bits	Function
D019	53273	7-4 3-0	Video matrix base address (inside VIC) Character dot-data base address (Bit = 1: IRQ occurred) VIC Interrupt Flag Register (bit = 1: IRQ occurred) Set on any enabled VIC IRQ condition Light pen-triggered IRQ flag Sprite-to-sprite collision IRQ flag Sprite-to-background collision IRQ flag Raster compare IRQ flag IRQ mask register: 1 = interrupt enabled Sprite-to-background display priority (1 = sprite) Sprites 0-7 multicolor mode select (1 = MCM) Sprites 0-7 expand 2x vertical (Y) Sprite-to-sprite collision detect Sprite-to-background collision detect Border color Background color 0 Background color 1 Background color 2 Background color 3 Background color 4 Sprite multicolor register 0 Sprite multicolor register 1 Sprite 0 color Sprite 1 color Sprite 2 color Sprite 3 color Sprite 4 color Sprite 5 color Sprite 6 color Sprite 7 color
D01A	53274		
D01B	53275		
D01C	53276		
D01D	53277		
D01E	53278		
D01F	53279		
D020	53280		
D021	53281		
D022	53282		
D023	53283		
D024	53284		
D025	53285		
D026	53286		
D027	53287		
D028	53288		
D029	53289		
D02A	53290		
D02B	53291		
D02C	53292		
D02D	53293		
D02E	53294		

MOS 6581 Sound Interface Device (SID)

Hex	Decimal	Bits	Function
D400	54272	7-4	Voice 1: Frequency control — low byte
D401	54273	3-0	Voice 1: Frequency control — high byte
D402	54274		Voice 1: Pulse waveform width — low byte
D403	54275		Unused
D404	54276		Voice 1: Pulse waveform width — high nibble
			Voice 1: Control register
		7	Select random noise waveform 1 = on
		6	Select pulse waveform 1 = on
		5	Select sawtooth waveform 1 = on
		4	Select triangle waveform 1 = on
		3	Test bit: 1 = disable oscillator 1
		2	Ring modulate osc. 1 with osc. 3 output 1 = on
		1	Synchronize osc. 1 with osc. 3 freq. 1 = on
		0	Gate bit: 1 = start att/dec/sus 0 = start release
D405	54277	7-4	Envelope generator 1: attack/decay cycle ctrl.
		3-0	Select attack cycle duration: 0-15
			Select decay cycle duration: 0-15

MOS 6510 I/O Registers

Hex	Decimal	Bits	Function
0000	0	7-0	MOS 6510 Data Direction Register (xx101111) Bit = 1: output Bit = 0: input x = don't care
0001	1		MOS 6510 on-chip I/O port /LORAM signal (0 = switch BASIC ROM out) /HIRAM signal (0 = switch kernal ROM out) /CHAREN signal (0 = switch char. ROM in) Cassette data output line Cassette switch sense (1 = switch closed) Cassette motor control (0 = on 1 = off) Undefined

MOS 6566 Video Interface Controller (VIC II)

Hex	Decimal	Bits	Function
D000	53248		Sprite 0 — X Pos
D001	53249		Sprite 0 — Y Pos
D002	53250		Sprite 1 — X Pos
D003	53251		Sprite 1 — Y Pos
D004	53252		Sprite 2 — X Pos
D005	53253		Sprite 2 — Y Pos
D006	53254		Sprite 3 — X Pos
D007	53255		Sprite 3 — Y Pos
D008	53256		Sprite 4 — X Pos
D009	53257		Sprite 4 — Y Pos
D00A	53258		Sprite 5 — X Pos
D00B	53259		Sprite 5 — Y Pos
D00C	53260		Sprite 6 — X Pos
D00D	53261		Sprite 6 — Y Pos
D00E	53262		Sprite 7 — X Pos
D00F	53263		Sprite 7 — Y Pos
D010	53264		Sprites 0-7 X Pos (msb of X coord)
D011	53265		VIC Control Register
		7	Raster compare: (bit 8) See 53266
		6	Extended color text mode: 1 = enable
		5	Bit-map mode: 1 = enable
		4	Blank screen to border color: 0 = blank
		3	Select 24/25-row text display: 1 = 25 rows
		2-0	Smooth scroll to Y dot-position (0-7) Read/write raster value for compare IRQ Light pen latch — X Pos Light pen latch — Y Pos Sprite display enable: 1 = enable VIC Control Register Unused
D012	53266	7-6	
D013	53267	5	ALWAYS SET THIS BIT TO 0!
D014	53268	4	Multicolor mode: 1 = enable (text or bit-map)
D015	53269	3	Select 38/40-column text display: 1 = 40 columns
D016	53270	2-0	Smooth scroll to X dot position (0-7) Sprites 0-7 expand 2x horizontal (Y) VIC Memory Control Register
D017	53271		
D018	53272		

NEW SOFTWARE for TRS 80 Model III and the Color Computer

■ Church Contribution System

designed to simplify and facilitate the tedious chore of recording envelopes. Provides a variety of reports. Maintains its own data-files. Only **\$150**

■ Data Base Manager

designed to help organize all your data and provide you with meaningful reports. Add or delete any information. New files can be created and old information transferred. Only **\$150**

■ Single Entry Ledger

designed as an uncomplicated control of finances for home or small business. Add, delete, edit at any time. Compatible with DBM. Only **\$95**

Write or phone for complete software price list.



Dept. MI 2
2457 Wehrle Drive
Amherst, NY 14221
716/631-3011

SeaFORTH for the Apple computer

Is a consistent structured operating system providing the advanced programmer with the tool to easily develop programs from machine language to high level compiled applications. With SeaFORTH, the edit-compile-execute-edit cycle is measured in seconds, not minutes.

The integrated SeaFORTH package includes:

- Editor
- Disc I/O
- Assembler
- Hi-res Graphics
- Transcendental Floating Point
- Command Line Input with Editing
- Detailed 150 Page Technical Manual with Complete Source Listing!

Implemented as a true incremental compiler, SeaFORTH generates machine code, not interpreted address lists. SeaFORTH's direct-threaded-subroutine implementation executes faster than interpreted address-list versions.

Apple SeaFORTH requires a 48K Apple II+, with DOS 3.3. Manual and copyable disk are available for only \$100.00

Compatible SeaFORTH for the AIM requires a terminal and is only available in EPROMs. Manual and EPROMs \$150.00

Manuals available, separately, for only \$30.00

All prices include UPS shipping.

VISA or MASTER CHARGE welcome.

(Dealer Inquiries Welcome)

TAU LAMBDA

P.O. Box 808, Poulsbo, Washington 98370
(206) 598-4863

Apple II+ and AIM are registered trademarks of Apple Computer and Rockwell

Advertiser's Index

Aardvark Technical Services, Ltd.	76
ABM Products	24
Amplify Inc.	62
Anthro-Digital Software	17
Apex Co.	24
Appletree Electronics	51
Ark Computing	12
Artsci, Inc.	IFC
Aurora Software Associates	83
Bedford Micro Systems	31
CGRS Microtech	63
Cleveland Consumer Computer Components	80
Collegiate Microcomputer	67
Commander	62
Compu Sense	49
CompuTech	28
Computer Mail Order	56-57
Computer Science Engineering	89
Computer Trader	99
Datamost, Inc.	34, 90, 92, IBC
Decision Systems	67
Digicom Engineering, Inc.	96
Digital Acoustics	84
D&N Micro Products, Inc.	21
Eastern House Software	39
Educational Computing Systems	10
Execom Corp.	40
Gimix, Inc.	1
Gooth Software	51
Hayden Software	36
Hudson Digital Electronics Inc.	68
Human Systems Dynamic	41
Interesting Software	95
Leading Edge	BC
Logical Devices	99
Lycor Computer	6
MICRObits (Classifieds)	53, 86, 87
MICRO INK	25, 31, 41, 44, 45
Micro Motion	28
Micro Signal	50
Micro-Spec, Ltd.	40
Micro-Ware Distributing Inc.	96
Midnight Software	49
MMS, Inc.	25
Modular Systems	83
Orion Software	18
Perry Peripherals	73
Privac, Inc.	2
Pterodactyl Software	105
Quentin Research	29
SGC	4
SJB Distributing	64
Skyles Electric Works	46, 58
Softel	72
Software Farm	33
Software Options	31
Southwestern Data Systems	106
Spectrum Systems	83
Spies Laboratories	43
Star Micronics	8
Tau Lambda	111
Unique Data Systems	102
Unique Software	96
Universal Data Research	111
Victory Software	20
XPS, Inc.	101

MICRO INK is not responsible for claims made by its advertisers. Any complaint should be submitted directly to the advertiser. Please also send written notification to MICRO.

Next Month in MICRO

January: Simulations/Applications/Math

- **Apple Math Editor** — This Apple Pascal program allows you to construct, edit, and print mathematical formulas easily.
- **Sun and Moon** — This Applesoft program produces a high-resolution graphic simulation of the apparent orbits of the sun and moon with respect to the Earth.
- **Measurement of a 35mm Focal Plane Shutter** — The program SHUTTER uses inexpensive hardware to measure the accuracy and repeatability of a focal plane shutter commonly found in 35mm cameras. Although written for the Atari 800, the program can be modified for any computer if you have access to three input pins, a ground, and the +5V power supply.
- **Methods to Evaluate Complex Roots** — A standard procedure to compute complex roots of polynomial equation.
- **Discrete Event Simulation on the Apple** — An explanation of techniques used in simulating real-world situations on a computer. An example program involving the flow of bank customers is presented.

Department Highlights

Apple Slices
PET Vet
From Here to Atari
CoCo Bits
Reviews in Brief
Software and Hardware Catalogs

Plus...

VIC Hi-Res Graphics Explained
Dealing with Atari's New Languages
Microcomputer Design of Transistor Amplifiers
More 68000 Instruction Set Tables

20% OFF

Your money goes farther when you subscribe. During the course of a year, when you subscribe, you save 20% (in the U.S.).

Pay only \$24.00 (\$2.00 a copy) for 12 monthly issues of MICRO sent directly to your home or office in the U.S.

More MICRO for Less Money When You Subscribe

But on the newsstand — if you can locate the issue you want — you pay \$30.00 a year (\$2.50 a copy).

Special Offer — Subscribe for 2 years (\$42.00) and get 30% off the single issue price.

Subscribe to MICRO today.

MICRO
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

Please send me MICRO for ☐ 1 year ☐ 2 years
NOTE: Airmail subscriptions accepted for 1 year only.

Check enclosed \$ _____
Charge my ☐ VISA account
☐ Mastercard account

No. _____

Expiration date _____

Name _____

Address _____

City/State _____ Zip _____

Subscription Rates Effective January 1, 1982

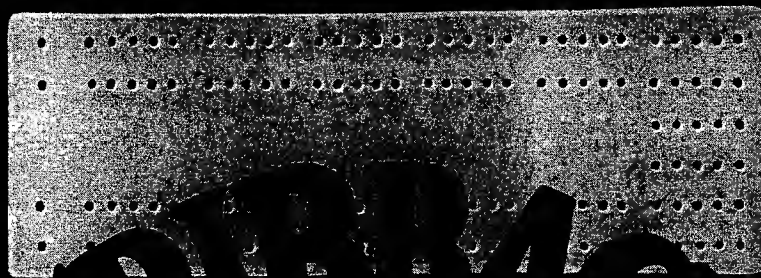
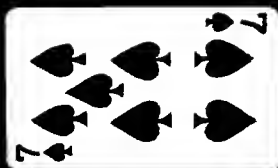
Country	Rate
United States	\$24.00 1 yr. 42.00 2 yr.
Foreign surface mail	27.00
Europe (air)	42.00
Mexico, Central America, Mid East, N. & C. Africa	48.00
South Am., S. Afr., Far East, Australasia, New Zealand	72.00

* Airmail subscriptions accepted for only 1 year.
For U.S. and Canadian 2-year rates, multiply by 2.

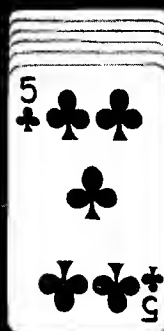
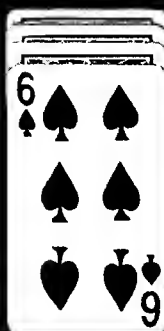
Job Title _____

Type of Business/Industry _____

By the Cardmaster—original Cribbage and 4 Solitaire games!



The Card Stars



When you're tired, upset, bored . . . in need of a challenge, or just relief from the ordinary there's nothing to compare with the fun and involvement of Solitaire or Cribbage. Within seconds you've forgotten the world and are absorbed in the play of the cards. And the brilliant way the Cardmaster programmed these games has taken out the effort and distractions but left in all the fun and challenge.

On a scale of 1 to 10, these card stars rate a big 11! . . . for sheer enjoyment and unmatched value. Think of it, five of the Cardmaster's best games at the price you'd expect to pay for just one! It includes original Cribbage, with your strategy against the Apple . . . plus 4 Solitaire games: Klondike, the all time standard, in 2 versions (1 or 3 cards at a time), exciting Picture Frame and the challenging Pyramid! If you or your family like cards at all, this is the one disk you must get!

Only \$34.95 for the Apple II* at your computer store or:

DATA MOST

9748 Cozycroft Ave., Chatsworth, CA 91311 (213) 709-1202.

VISA/MASTERCARD accepted. \$2.00 shipping/handling charge.
(California residents add 6 1/2% sales tax.)

*Apple II is a trademark of Apple Computer, Inc.

THE PROWRITER COMETH.

(And It Cometh On Like Gangbusters.)



Evolution.

It's inevitable. An eternal verity.

Just when you think you've got it knocked, and you're resting on your laurels, somebody comes along and makes a dinosaur out of you.

Witness what happened to the Centronics printer when the Epson MX-80 came along in 1981.

And now, witness what's happening to the MX-80 as the ProWriter cometh to be the foremost printer of the decade.

SPEED

MX-80: 80 cps, for 46 full lines per minute throughput.

PROWRITER: 120 cps, for 63 full lines per minute throughput.

GRAPHICS

MX-80: Block graphics standard, fine for things like bar graphs.

PROWRITER: High-resolution graphics features, fine for bar graphs, smooth curves, thin lines, intricate details, etc.

PRINTING

MX-80: Dot matrix business quality.

PROWRITER: Dot matrix correspondence quality, with incremental printing capability standard.

FEED

MX-80: Tractor feed standard; optional friction-feed kit for about \$75 extra.

PROWRITER: Both tractor and friction feed standard.

INTERFACE

MX-80: Parallel interface standard; optional serial interface for about \$75 extra.

PROWRITER: Parallel and serial interface standard.

WARRANTY

MX-80: 90 days, from Epson.

PROWRITER: One full year, from Leading Edge.

PRICE

Heh. heh.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021. Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

LEADING EDGE.

For a free poster of "Ace" (Prowriter's pilot) doing his thing, please write us.